Christian Lemaître
Carlos A. Reyes
Jesús A. González (Eds.)

LNAI 3315

Advances in Artificial Intelligence – IBERAMIA 2004

9th Ibero-American Conference on AI
Puebla, México, November 2004
Proceedings

Springer

VISIT…

LANZAROTE
Caliente.COM

Lecture Notes in Artificial Intelligence 3315

Edited by J. G. Carbonell and J. Siekmann

Subseries of Lecture Notes in Computer Science

Christian Lemaître   Carlos A. Reyes
Jesús A. González (Eds.)

Advances in Artificial Intelligence – IBERAMIA 2004

9th Ibero-American Conference on AI
Puebla, México, November 22-26, 2004
Proceedings

Springer

eBook ISBN: 3-540-30498-3
Print ISBN: 3-540-23806-9

Preface

The 9th Ibero-American conference on Artificial Intelligence IBERAMIA 2004 took place in Mexico for the third time in 16 years, since the first conference organized in Barcelona in January 1988. It was also the second time that the conference was held in the state of Puebla. The first time, in 1996, it was the Universidad de la Américas Puebla that was in charge of the local organization of the conference, this year it was the turn of the Instituto Nacional de Astrofísica, Óptica y Electrónica, INAOE, to do it.

The 1996 conference was the last conference where all the papers were presented in Spanish or Portuguese. Since then the proceedings have been published in English by Springer in the LNAI series. This linguistic change was a sign of the scientific maturity of the Ibero-American artificial intelligence community and the best way for it to share with the international artificial intelligence community the best results of many of its research groups. It was also the way to open this forum to researchers of other countries to enrich the scientific content of the conferences.

One relevant feature of the last four conferences with the proceedings published in English by Springer is that, besides the participation of people from many countries, the majority of papers came from Ibero-American researchers. We can state that IBERAMIA has consolidated itself as the main scientific forum where the Ibero-American artificial intelligence researchers meet together every other year. In 2004 we received 304 papers, 97 of which were accepted; this comes up to an acceptance rate of 31%. The figures are similar to those of the 2002 Sevilla conference with 316 received papers and 97 accepted papers. The numbers of submitted and accepted papers per country are shown in the following table:

Country	*Submitted*	*Accepted*	*Country*	*Submitted*	*Accepted*
Argentina	3	2	India	2	0
Austria	4	0	Iran	4	1
Belgium	1	1	Israel	1	0
Brazil	54	18	Korea	12	2
Canada	4	1	Mexico	103	28
Cuba	6	1	Portugal	17	5
Chile	6	4	Spain	56	26
China	2	1	Tunisia	4	1
France	7	1	USA	2	2
Germany	2	0	Venezuela	12	3
UK	2	0			

The AI topics covered by the submitted and accepted papers can be seen in the following table:

Topic	*Submitted*	*Accepted*
Distributed artificial intelligence and multi-agent systems	28	7
Knowledge engineering and case-based reasoning	21	4
Planning and scheduling	18	8
Machine learning and knowledge acquisition	23	6
Natural language processing	34	8
Knowledge representation and reasoning	20	10
Knowledge discovery and data mining	23	4
Robotics	24	8
Computer vision	32	13
Uncertainty and fuzzy systems	11	4
Genetic algorithms and neural networks	45	15
AI in education	14	4
Miscellaneous topics	11	5
Total	304	97

IBERAMIA 2004 was organized as an initiative of the Executive Committee of IBERAMIA. This committee is in charge of the planning and supervision of IBERAMIA conferences. Its members are elected by the IBERAMIA board which itself is made up of representatives from the following Ibero-American associations: AEPIA (Spain) APPIA (Portugal), SBC (Brazil), SMIA (Mexico). This book contains revised versions of the 97 papers selected by the program committee for presentation and discussion during the conference. The volume is structured into 13 thematic groups according to the topics addressed by the papers.

Acknowledgements

We would like to express our sincere gratitude to all the people who helped to bring about IBERAMIA 2004. First of all thanks to the contributing authors, for ensuring the high scientific standard of the conference and for their cooperation in the preparation of this volume.

Special thanks are due to the members of the program committee and auxiliary reviewers for their professionalism and their dedication in selecting the best papers for the conference. Thanks also to the IBERAMIA Executive Committee for their guidance and their continuous support.

We owe particular gratitude to the invited speakers for sharing with us their experiences and their most recent research results.

Nothing would have been possible without the initiative and dedication of the Organizing Committee, and the support of INAOE. We are very grateful to all the people who helped in the large variety of organizing tasks, namely Hector Lopez our web manager, Gabriela López Lucio and Luis Villaseñor Pineda our publicity managers, Josué Pedroza for his great job with the management of the CyberChair system during the submission and evaluation processes, Jesus A. Gonzalez, Oscar E. Romero A. and Ivan Olmos for their help in the preparation of this book, Angélica Muñoz, Guillermo de Ita, and Olac Fuentes, for their contribution to the management of the tutorials and workshops, Gorgonio Cerón Benítez and Carmen Meza Tlalpan for their help in the management of local arrangements and financial issues, Dulce Millan and Nidia Lara for their useful support in the administrative duties, and Lupita Rivera for the contacts with the media. All the members of the local committee headed by Carlos Alberto Reyes did a great job.

Thanks to the invited speakers, the tutorial instructors and workshops chairs for giving more relevance to the Conference General Program.

The French-Mexican Laboratory of Informatics, LAFMI, supported part of the travel expenses of the Program Chair between Xalapa and Tonantzintla.

We would like to thank the Benemérita Universidad Autonoma de Puebla, for its support of the inaugural session and first invited speech held in the beautiful and historical conference hall Salon Barroco. We want to thank also the Universidad de la Américas Puebla for their logistics support during the conference. We are also grateful to Microsoft Mexico, and especially to Luis Daniel Soto, for its financial support and for its contribution of an invited speaker and a tutorial. Our gratitude to Francisco Soto, Research and Graduate Studies Director of INAOE, and Aurelio López, Head of the Computer Science Department of the INAOE, for their continuous support throughout this year.

Tonantzintla, Puebla,
November 2004

Christian Lemaître
Program/Chair

Carlos A. Reyes
Organization/Chair

Jesus A. Gonzalez
Cyber/Chair

This page intentionally left blank

IBERAMIA 2004 Organizing Committee

Program and Scientific Chairman

Christian Lemaître
Mexico

Organization Chairman

Carlos Alberto Reyes García
INAOE, Mexico

Steering Committee

Christian Lemaître, LANIA, Mexico
Alvaro de Albornoz, SMIA, Mexico
Arlindo Oliveira, APPIA, Portugal
Federico Barber, AEPIA, Spain
Francisco Garijo, Telefónica I+D, Spain
Helder Cohelo, University of Lisbon, Portugal
Jaime Sichman, SBC, Brazil
Miguel Toro, University of Sevilla, Spain

Program Committee

Abraham Sánchez
Agostino Poggi
Alejandro Ceccatto
Alexander Gelbukh
Alexis Drogoul
Alberto Oliart Ros
Amal El Fallah Seghrouchni
Ana García Serrano
Ana Teresa Martins
Analia Amandi
Andre Ponce de Leon F. de Carvalho
Andrés Pérez Uribe
Angel P. del Pobil
Anna Helena Reali Costa
Antonio Bahamonde
Antonio Ferrandez
Antonio Moreno
Ariadne Carvalho
Arlindo Oliveira
Arturo Hernández Aguirre
Beatriz Barros
Bob Fisher
Carlos A. Brizuela
Carlos A. Coello Coello
Carlos Alberto Reyes García
Carolina Chang
Celso A. Kaestner
Chilukuri K. Mohan
Dibio Leandro Borges
Duncan Gillies
Ed Durfee
Eduardo Morales

Additional Reviewers

Aida Valls
Akiko Inaba
Alejandro Zunino
Alessandro Lameiras
Alexandre da Silva
Alexandru Suna
Alicia Troncoso
Aloisio Carlos de Pina
Amanda Smith
Amit Bhaya
Ana Carolina Lorena
Ana Paula Rocha
Andreia Grisolio M.
Andrew Coles
Antonio Fernández
Antonio Garrido
Antonio Lova
Armando Matos
Armando Suárez
Arnaldo Mandel
Aurelio López López
Bernhard Peischl
Brahim Hnich
Carla Koike
Carlos Brito
Carlos Castillo
Carlos Hitoshi Morimoto
Cedric Pradalier
Charles Callaway
Daniel Koeb
David Allen
David Pearce
Derek Long
Edgardo Vellón
Efren Mezura-Montes
Elie Chadarevian
Elizabeth Tapia
Emmanuel Mazer
Everardo Gutierrez
Fabiola Lópes y López
Fabrício Enembreck
Federico Ramírez Cruz
Fernando Carvalho
Fernando Godínez D.
Fernando Llopis
Fernando López
Francine Bicca
Francisco Ferrer
Francisco Rodríguez
Fredric Marc
Giordano Cabral
Gustavo Batista
Gustavo Olague
Hae Yong Kim
Heidi J. Romero
Hiram Calvo-Castro
Huei Diana
Hugo Jair Escalante
Hugo Santana
Ignacio Mayorga
Ivan Olmos Pineda
Ivana Sumida
Jacques Robin
Jacques Wainer
Javier Giacomantone
Javier Martínez-Baena
Jerónimo Pellegrini
Jesús Peral
Joaquim Costa
Joerg Muller
John Lee
Jos Alferes
José A. Garcia
José A. Troyano
José M. Puerta
José Palma
Juan Antonio Navarro
Juan Carlos López
Julio Cesar Nievola
Kamel Mekhnacha
Karina Valdivia Delgado
Keith Halsey
Ligia Ferreira
Louise Seixas
Luis Berdun
Luis C. González
Luis Damas
Luís Filipe Antunes
Luis Miguel Rato
Luis Paulo Reis
Luis Sarmento
M. Carmen Aranda
Manuel Chi
Manuel Mejia Lavalle
Manuel Montes-y-Gómez
Marcelino Pequeno
Marcello Balduccini
Marcelo Andrade T.
Marcelo Armentano
Marco Aurelio Pacheco
Marcos Cunha
Marc-Philippe Huget
Mats Petter Pettersson
Michel Ferreira
Michele Tomaiuolo
Miguel A. Salido
Miguel Arias Estrada
Mikal Ziane
Nejla Amara
Nelma Moreira
Nick Campbell
Nicolas Sabouret
Nik Nailah Abdullah
Oliver Obst
Olivier Lebeltel P.
Orlando Lee
Pablo Granitto
Pablo Verdes
Paola Turci
Patricia A. Jaques
Peter Gregory
Pilar Tormos
Rafael Muñoz
Raúl Giráldez
Reinaldo A.C. Bianchi
Renata Vieira
Ricardo Azambuja S.
Ricardo Bastos C.
Ricardo Martins de A.
Ricardo Silveira
Riverson Rios
Robinson Vida

Rolando Menchaca M.
Ronaldo Cristiano P.
Rosa Rodríguez S.
Samir Aknine
Sandra Alves
Silvia Schiaffino
Silvio do Lago P.
Solange Oliveira R.
Steve Prestwich
Thamar Solorio Martínez
Timothy Read
Trilce Estrada Fiedra
Valdinei Preire
Valguima Odakura
Victoria Eyharabide
Vilma França Fernandes
Vitor Beires Nogueira
Vladik Kreinovich
Xavier Alaman
Xavier Blanc
Xose R. Fdez-Vidal
Yichen Wei
Yingqian Zhang

Table of Contents

Checking Social Properties of Multi-agent Systems with Activity Theory

Rubén Fuentes, Jorge J. Gómez-Sanz, and Juan Pavón

Universidad Complutense Madrid, Dep. Sistemas Informáticos y Programación, 28040 Madrid, Spain*
{ruben, jjgomez, jpavon}@sip.ucm.es
http://grasia.fdi.ucm.es

Abstract. Many approaches of the agent paradigm emphasize the social and intentional features of their systems, what are called *social properties*. The study of these aspects demands their own new techniques. Traditional Software Engineering approaches cannot manage with all the information about these components, which are as related with software development as with social disciplines. Following previous work, this paper presents a framework based in the Activity Theory to specify and verify social properties in a development process for multi-agent systems. Using this framework developers acquire tools for requirements elicitation and traceability, to detect inconsistencies in their specifications, and to get new insights into their systems. The way of working with these tools is shown with a case study.

Keywords: Multi-agent Systems Development, Activity Theory, Validation and Verification.

1 Introduction

Multi-Agent Systems (MAS) are usually conceived as organizations of autonomous and rational entities that work together to achieve their common goals. This perspective implies that aspects of organization, cognition, development, and motivation have to be deeply study. However, existing MAS methodologies usually simplify the analysis of these *social properties* to a problem of defining roles and power relationships (as it is the case for INGENIAS [14] or KAOS [2]). Since social features comprehend a richer set of features, they demand additional theoretical background, abstractions, and techniques.

As a source for this required knowledge, we have considered research in social sciences, concretely the Activity Theory. The Activity Theory (AT) [10] is a cross-disciplinary framework for the study of human doings embedded in its socio, cultural, and historical context. It considers that the social and individual levels of human

* This work has been funded by Spanish Ministry of Science and Technology under grant TIC2002-04516-C03-03.

activities are intrinsically interleaved. Besides, the social component also includes the historical development of those activities as the experience collected by the society carrying them out. Our previous work introduced an approach to model some social aspects of MAS with the AT. This approach includes a UML language for AT concepts and processes to use its techniques in MAS development. In [3] we used the AT as a basis to detect contradictions in the MAS design process. In [5], a similar AT based schema was the tool to identify requirements. The evolution of this work takes us to consider that *requirements* and *contradictions* can be seen as particular cases of social properties. Consequently, AT is suggested as an appropriate framework to address the use of *social properties* in MAS development.

In the AT framework for MAS, *social properties* are described through patterns that include a textual explanation and a diagram in the UML language for AT. The properties have a set of match patterns, which allow its detection, and a set of solution patterns, which suggest modifications in the models. The application of these patterns to a concrete methodology is possible thanks to the use of mappings. Mappings specify how to translate concepts from AT to those in the MAS methodology. These correspondences allow the applicability of this approach in different MAS methodologies and a semi-automated method to work with social properties.

So far, we have classified social properties in three types according to their role in development. These properties can represent configurations of the system that developers have to preserve (e.g. requirements), or to avoid (e.g. contradictions in the information), or knowledge about the MAS (e.g. the existing organization).

The remaining paper has six additional sections. Section 0 discusses about the need of new techniques to cope with the use of social properties inherent to the agent paradigm. Then, section 0 explains the way of describing social properties in terms of the UML language for AT concepts, while section 0 gives some examples of these properties in the three identified categories. Section 0 describes a method, which can be automated, to use the social properties in the validation and verification of models. Section 0 shows the use of this method with a case study on a real specification. Finally, the conclusions discuss the results obtained in the validation and verification of social and intentional properties of MAS with this approach.

2 The Need of Tools for Social Features in MAS

Software Engineering involves a continuous research about new concepts and methodologies that make possible building more complex software systems. One of the main advantages of the agent paradigm is that it constitutes a natural metaphor for systems with purposeful interacting agents, and this abstraction is close to the human way of thinking about our own activities [11]. This foundation has lead to an increasing interest in social sciences (like in the works of [2], [11], and [15]) as a source for new concepts and methods to build MAS. However, this interest has hardly covered some interesting social features of the MAS design. Here, the term *"social features"* encompasses organization culture, politics, leadership, motivation, morale, trust, learning, or change management. There are two main reasons in MAS research to give a novel and special attention to these features: the human context and the own essence of the agent abstractions.

Firstly, the environment of a software system, defined as the real world outside it, is usually a human activity system [13]. Its study must consider then the *social features* of humans and their societies. Software Engineering research in branches like Requirements Engineering [1], CSCW [6], or HCI [9], already makes an extensive use of social disciplines to grasp the relevant information about the human context.

Besides the human context, MAS are modelled in a very alike fashion to human organizations, as societies of collaborating intentional entities [11], [16]. It gives a possibility of describing some properties of the system and its context, the social and intentional ones, in quite a uniform way, taking advantage of the knowledge extracted from human sciences.

These reasons take us to consider the need of generic mechanisms to model these social properties of the MAS and to validate the specifications against them. These properties do not appear in traditional software methodologies. If MAS design has to be a new level of abstraction for system design, it needs to develop innovative tools that consider these social concepts. Previous experience with the description and checking of social properties about contradictions [3] and requirements [5], take us to propose the use of the AT for this purpose.

3 Describing *Social Properties* with AT

Before considering what the better manner of representing a social property for development is, we must consider that it can represent several types of information for that development; it can correspond to a configuration to keep, i.e. a pattern property, or to avoid, i.e. an anti-pattern property, or a feature to discover, i.e. a descriptive property. In the case of a pattern property, a problem with it occurs when there is no match with the pattern or just a partial one. For configurations to avoid, problems arise when there is a complete match. The descriptive properties help developers to discover new information about a given system, so there are no conflictive situations inherently related with their absence or presence.

According to these possible roles of social properties in MAS development, their representation has to fulfil three aims. Firstly, the properties have to be a tool for the development. Their description should be in a language understandable by developers and suitable for automated processing. Secondly, the representation should build a common language between customers and developers. The development of a system is a joint venture of people with very different backgrounds, what makes difficult reciprocal understanding. AT vocabulary uses social abstractions, which are close to both customers and developers and therefore should facilitate their mutual understanding [1]. Finally, the representation of properties should help to solve the problems related with themselves, such as the non-accomplishment of requirements or the appearance of contradictions.

To satisfy these requirements, social properties are represented with two components: a set of match patterns and other of solution patterns. Each pattern of these components has two possible representations, a textual form and another one based on the use of UML stereotypes to represent AT concepts [3]. The UML form is the basis for the automated process of pattern detection and problem solving. In this process, the stereotypes and names of the UML form can be variables or fixed values. This allows fixing some values of the properties before the detection procedure and combining patterns through shared values. On the other hand, the textual form is intended to give further information about the patterns. It helps customers to understand the meaning of the used UML notation and enables both customers and developers to know the social interpretation of the pattern.

According to our description, a social property can include two different sets of patterns: one for detection and other for solution. A match pattern describes a set of entities and their relationships, which represents the property. It acts as a frame that has to be instantiated with information from the specification. If a set of artefacts in the specification fits into the pattern, the property is satisfied. In the case of both properties to keep and properties to avoid, the solution pattern is a rearrangement of its corresponding match pattern, maybe with additional elements. Solution patterns can correspond to partial or full matches. For the total absence of match, there is no point in defining a solution.

It is remarkable to note that patterns for detection and solution are not tied in fixed pairs; moreover, they can be reused or combined through shared variables to describe new situations. An example of this possibility can be found in the case study described in section 0, which combines match patterns to describe situations that are more complex than originally.

4 Social Properties for MAS

This section presents examples of properties used in MAS development for every type previously identified and gives their representation. The first one corresponds to a pattern property that is a requirement describing a social setting in the system environment, and it is adapted from the Activity Checklist [9]. The second one is an anti-pattern property describing a contradiction from the AT, the Exchange Value contradiction extracted from [8]. Finally, there is a descriptive property about a hierarchical organization as described in [16]. In the present section, the textual form of the properties is included as part of the explanation of the diagrams. Words in *italics* represent concepts from the AT vocabulary.

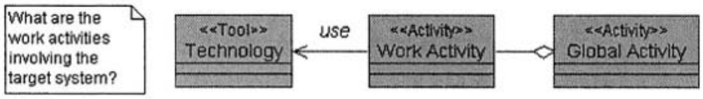


Fig. 1. A question about the system context

The Activity Chec7klist [9] is an analytical tool to elicit contextual knowledge about an *activity*. It is composed by a set of aspects that have questions expressed in terms of natural language to grasp their information. One of these aspects is concerned with the role that current technology plays in producing the desired *outcomes*. A related question with this aspect is “What are the work activities involving the target system?”. Fig. 1 shows that the relevance of a *tool* in the organization is given by the *activities* in which it participates. The study of these activities helps developers and customers to assess the importance of the given technology.

The second example adapts the Exchange Value contradiction described by [8]. In a MAS, this contradiction emerges when a *subject* has to generate a *product* to be consumed by other members of the *community*. However, none of the generator *subject*’s *goals* are satisfied by that *product* or he is not motivated enough to do it, for example because the task has also negative effects. Consequently, the *subject* that is able to create the *product* and give it to the *community* refuses to do it. The left side of Fig. 2 illustrates a version of this contradiction in which the agent does not have an *objective* related to the creation of the *outcome*, and for that reason the task never gets executed. Therefore, the members of the *community* are unable to satisfy their needs. The transition of the required product, i.e. *Outcome 1*, from *outcome* in *Activity 1* to *artifact* in *Activity 2* is indicated with the relation *“change of role”*.

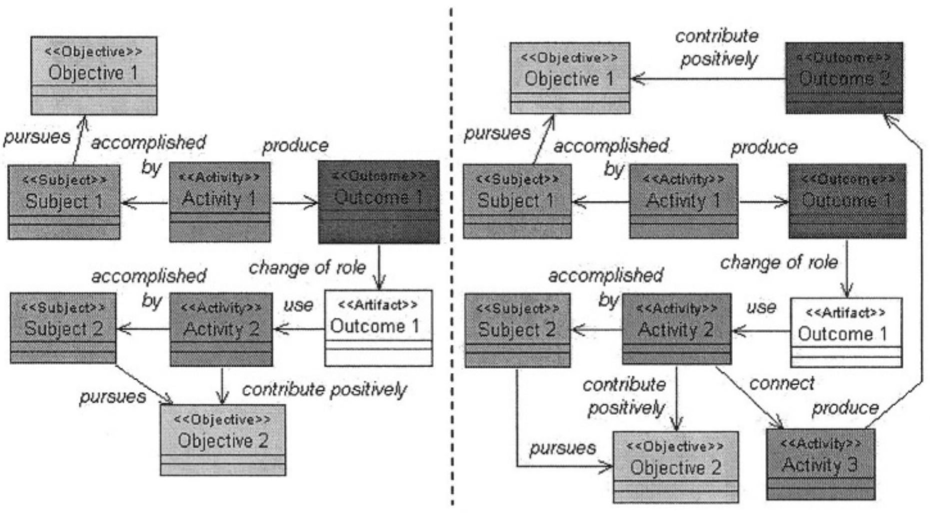


Fig. 2. The Exchange Value contradiction on the left and a possible solution on the right

A possibility to overcome the Exchange Value contradiction is to encourage the *subject* by giving him a reward for executing the task. The *subjects* that benefit from the product of the first *subject* should provide him with products that satisfy one of its needs. The model on the right of Fig. 2 shows this new situation.

The final example is an identification of the social structures in MAS. In the overview about MAS by [16], several types of possible organizations for the community of agents were identified. A *hierarchy* was defined as an organization where “The authority for decision making and control is concentrated in a single problem solver [...]. Superior agents exercise control over resources and decision making.". This situation can be described with Fig. 3. The *SuperiorAgents* can give orders that *SubordinatedAgents* have to accomplish, that is, *SuperiorAgents* are able to generate new *objectives* for their *Subordinated Agents.* The relationships labelled *"change of role"* represents that the *outcome* of the *activity* that generates the orders, i.e. the *Objective,* becomes an *objective* for the *SubordinatedAgent.*

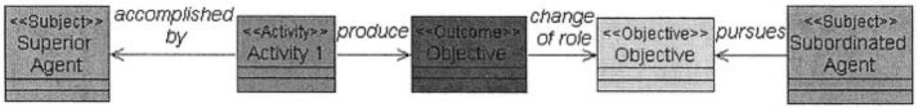


Fig. 3. A hierarchical organization in a MAS

Detecting this kind of descriptive-properties about the MAS gives developers information about how is their system. This knowledge can help them to decide the best design solution to challenges in their systems. Examples of this are ways to solve negotiation problems in a MAS according to its organizational structure.

5 Checking a Specification with Social Properties

This section introduces a method to check social properties against a specification. It is a generalization of the one used for AT contradictions [4].

The method needs three parameters:

- *Mappings.* They allow translations between the concepts of AT and the given agent oriented methodology. In this way, the method becomes generic as it could be applied to any agent-oriented methodology, without demanding a vocabulary based on AT concepts. A more detailed description on how to build these mappings and an example with the INGENIAS methodology [14] can be found in [4].
- *Social properties to verify.* They are described as shown in preceding sections. They can be predefined, e.g. requirements [5] or contradiction patterns [3] of the AT, or defined by users. The process itself can then be regarded as "validation", i.e. when its patterns are requirements, and "verification", i.e. when its patterns represent other kind of properties.
- *MAS to check.* Since the process uses mappings, there is no prerequisite about the language of the specification as long as it based in the agent paradigm.

The checking process itself includes the following steps:

1. Translate the MAS specifications to the AT language with the mappings. Translation is not a trivial task since correspondences between structures and their translations are usually relations "many to many". So, the translated structure needs to keep a reference to the original one.

2. For every property, look for correspondences of its match patterns in the specification. The process of properties detection is one of pattern matching. Models are traversed seeking groups of elements with the same structure and slot values that match patterns. When a corresponding structure is found, the property is considered as satisfied. Partial matches are also possible as they can represent a conflictive situation, for example a requirement which is not preserved.
3. For a matching in the models, propose the customized solution pattern. A match pattern, or a part of it, can have a related solution pattern. This pattern describes a change in the models to solve a problem or enhance some aspect of the specifications. Typically, these solutions are rearrangements of the elements involved in the match pattern where additional elements can be involved.

This method can be semi-automated, but user interaction is still needed to determine values for the variables in patterns, decide when a match makes real sense, or the best manner to modify models.

The main advantages of the overall approach over others to import social sciences in Software Engineering (like those in [1], [6], and [12]) are that this proposal uses UML, which is well known for developers and more adequate for users than formal languages, and it provides a structured method to work with its social properties.

6 Case Study

In order to show how to apply the social properties and the checking process, this paper presents a case study based on agent teams in the programming environment *Robocode* [7]. *Robocode* simulates tank battles through robots that actuate with predefined primitives (like ahead, turn left, and fire) and perceive the situation with position and radar sensors and the detection of some events. Developers have to program the behaviour of those tanks to destroy their enemies and survive the battle.

The proposed case study considers collaboration between tanks in this frame. It is modelled with the INGENIAS methodology [14] and its full specification can be found at *http://ingenias.sourceforge.net.* The case considers armies of collaborative tanks composed by squadrons. In every squadron, agents can play one of two roles:

- *Soldiers.* These are the basic members of the squadron. They try to survive the battle while destroying their enemies and accomplishing the orders of their *captain.*
- *Captains.* They are the leaders of the squadrons. Showing the basic performance of a soldier, they a captain also plans a strategy for its soldiers, communicates it to them, tracks the course of the battle, and makes modifications to the planning if needed. A squadron can have just one *captain* but several *soldiers.*

The previous situation is summarized in Fig. 4. The elements involved are: *DynamicArmy,* which is an organization; *DynamicGroup,* which is a group of tanks; *TeamAgent, TeamMate* and *TeamLeader,* which are agents; *Soldier* and *Captain,* which are roles; circles represent goals. A *TeamLeader* pursues *CommandSoldiers.*

The task to satisfy this objective generates and communicates orders to the *captain*'s troops. A *TeamMate* of these troops tries to satisfy the goal *AccomplishOrders.* Thus, it tries to obey the orders of its *TeamLeader.* A common goal for all the agents, which is inherited from the *TeamAgent,* is *SurviveBattle.* This goal forces the agents to preserve their life, whatever the situation can be.

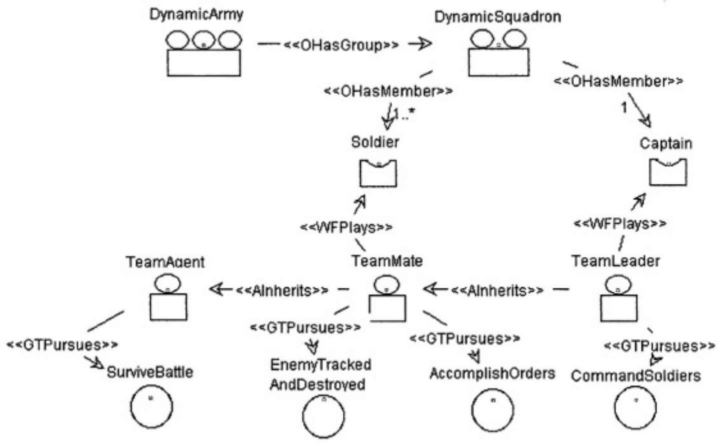


Fig. 4. Specification of a *Robocode* army with INGENIAS

A possible strategy for the squadron is to have scouts that go forward and backward from its lines in order to determine the position of enemy troops. With this information its leader can plan an attack in a very precise way. The problem is that this is a high-risk task for the agent who accomplishes it, which contradicts its main objective of *SurviveBattle.* Fig. 5 represents this situation where *GeneratePlan* is a task for a *TeamLeader* and *Explore* is for a *TeamAgent.*

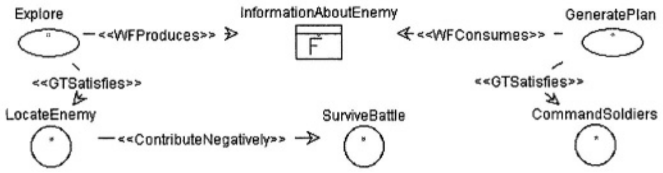
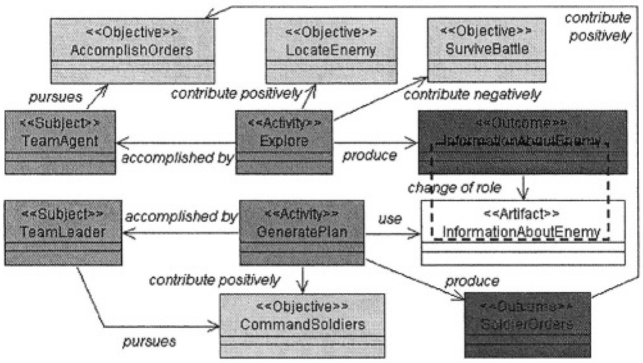


Fig. 5. Tasks, goals, and mental entities involved in the explore strategy

Since *Explore* is a task that contributes negatively to a compulsory goal, i.e. *SurviveBattle,* the *TeamAgent* can avoid its execution for the sake of other objectives. However, this task is vital to the global strategy of its squadron. This situation corresponds to a variant of the Exchange Value Contradiction that can be seen in Fig. 6 if suppressing the *SoldierOrders* entity.

Given the Exchange Value contradiction, the solution pattern previously introduced would suggest the addition of a new *activity* to the *TeamLeader,* which would allow him to reward the *TeamAgent.* Nevertheless, this is not necessary, provided the fact that the *TeamLeader* and the *TeamAgent* are embedded in a *Hierarchy* pattern (presented in Fig. 3), which enables the *TeamLeader* to solve the contradiction through commands without additional activities. The solution appears in Fig. 6 where the entities in the rectangle are the part added to the exchange value contradiction because of the of the hierarchy pattern.



<<Objective>> AccomplishOrders
<<Objective>> LocateEnemy
<<Objective>> SurviveBattle
contribute positively
pursues
contribute positively
contribute negatively
<<Subject>> TeamAgent
<<Activity>> Explore
<<Outcome>> InformationAboutEnemy
accomplished by
produce
change of role
<<Subject>> TeamLeader
accomplished by
<<Activity>> GeneratePlan
use
<<Artifact>> InformationAboutEnemy
contribute positively
produce
<<Objective>> CommandSoldiers
<<Outcome>> SoldierOrders
pursues

Fig. 6. Exchange Value contradiction and its solution in a hierarchy

This final graphical model in Fig. 6 deserves some careful considerations. First, the combined use of several patterns has allowed building a complex pattern and an innovative solution different to the standard with a new activity. The second observation is that the solution does work because of a qualitative arithmetic of goals. The *TeamAgent* has to know that *AccomplishOrders* is even more relevant than *SurviveBattle;* otherwise, a different conflict between contradictory objectives arises.

Independently of the solution adopted, the final decisions about the adequacy of the match, the proposed solution, and other patterns always require human judgement.

7 Conclusions

This paper gives a novel approach to manage social properties in MAS specifications. Social properties are related with the most specific components of the agent paradigm, i.e. its social and intentional features. Social sciences have studied this kind of features for a long time in human societies and can provide useful insights in them. The Agent Oriented Software Engineering can profit of this knowledge to improve its own understanding about these aspects and create new techniques that works with the complex interactions between their systems and the surrounding human environment.

Following previous work [3], [4], [5], this proposal uses the Activity Theory as the foundation to study these social properties. The knowledge of the AT has crystallized in several tools for the MAS development: a way to define social properties, a process to check them in MAS specifications, and a library of social properties.

The description of social properties with two representations, one with natural language and other with UML, tries to make them understandable for both customers and developers. Customers know the real domain of the system and have to communicate that information to the developers that use it in the implementation. Of course, the used language does not involve that subjacent concepts are understood. Here, AT takes advantage of the fact that its concepts are grounded in common knowledge about our own human societies, and then, it can be easily internalised by all the members in the development team.

To verify the social properties in a MAS development, this paper describes a method independent of the considered MAS methodology. The process uses mappings to translate specifications between the AT and the MAS language. In this way, developers do not need to learn a new methodology and can use their own tools.

The third element of our approach is the set of libraries that collects information from AT studies and MAS projects. These libraries are repositories of predefined social properties that developers can use in their projects. Currently, there are two of these libraries available: one for requirements based in the Activity Checklist with twenty properties now; and other for contradictions according to AT research, which includes ten properties nowadays. These repositories are integrated with the INGENIAS Development Kit as a proof of the feasibility of the overall approach.

This ongoing research has three main open issues in its current status. The first one is about the need of interactive work with the tools. Users have to decide what properties to check, how they have to be customized, judge their meaning in their specifications, and select the better way of modifying the system. Although user's judgement will always be necessary, his workload can be reduced with more detailed patterns and increased reasoning capabilities in the checking method. The second issue is related with the expressive power of the UML language for AT. This language has to support the translation of the knowledge from sources in the very rich natural language. Our UML notation cannot support all the features of the natural language but should include a proper set of primitives to transmit the key meanings of the social properties and allow the automated processing at the same time. Finally, the enrichment of the properties libraries is also work to do.

References

1. Bødker, S., Grønbœk, K.: *Cooperative prototyping: users and designers in mutual activity.* International Journal of Man-Machine Studies 34 (3): pp. 453-478. 1981.
2. A. Dardenne, A. van Lamsweerde, S. Fickas: *Goal-directed Requirements Acquisition.* Science of Computer Programming, Vol. 20, 1993, pp. 3-50.
3. R. Fuentes, J.J. Gómez-Sanz, J. Pavón: *Activity Theory for the Analysis and Design of Multi-Agent Systems.* Proceedings of the 4th International Workshop on Agent Oriented Software Engineering (AOSE 2003). Vol. 2935 of LNCS, pp.110–122. Springer Verlag. 2003

4. R. Fuentes, J.J. Gómez-Sanz, J. Pavón.: *Social Analysis of Multi-Agent Systems with Activity Theory.* Proceedings of CAEPIA 2003, San Sebastian, Spain, November 2003. Vol. 3040 of LNAI. Springer Verlag. 2004.
5. R. Fuentes, J.J. Gómez-Sanz, J. Pavón.: *Towards Requirements Elicitation in Multi-Agent Systems.* Proceedings of the 4th International Symposium From Agent Theory to Agent Implementation (AT2AI 2004), Vienna, Austria, April 2004.
6. Hughes, J., King, V., Rodden, T., Andersen, H.: *Moving out from the control room: ethnography in system design.* Proceedings of the ACM 1994 Conference on Computer Supported Cooperative Work - CSCW'94. ACM Press, pp. 429-439.
7. IBM alphaWorks: *Robocode.* 2002. http://robocode.alphaworks.ibm.com
8. E. V. Ilyenkov: *The dialectics of the abstract and the concrete in Marx's Capital.* Moscow: Progress. 1982.
9. V. Kaptelinin, B. A. Nardi, C. Macaulay: *The Activity Checklist: A tool for representing the "space" of context.* Interactions, 6 (4), pp. 27-39. 1999.
10. A. N. Leontiev: *Activity, Consciousness, and Personality.* Prentice-Hall. 1978.
11. Pattie Maes: *Modeling Adaptative Autonomous Agents.* Artificial Life Journal, C. Langton, ed., Vol. 1, No. 1 & 2, MIT Press, 1994.
12. G. M. McGrath, L. Uden: *Modelling Softer Aspects of the Software Development Process: An Activity Theory based approach.* Proceedings of the HIemCSS-33 - Software Process Improvement. IEEE Computer Society Press. Jan. 2000.
13. B. Nuseibeh, S. Easterbrook: *Requirements Engineering: A Roadmap.* Proceedings of the 22nd International Conference on Software Engineering (ICSE-2000). ACM Press. 2000.
14. Pavón J., Gómez-Sanz, J.: *Agent Oriented Software Engineering with INGENIAS.* Proceedings of the 3rd International Central and Eastern European Conference on Multi-Agent Systems (CEEMAS 2003). Vol. 2691 of LNCS, pp. 394-403. Springer Verlag. 2003
15. J. Sichman, Y. Demazeau: *On Social Reasoning in Multi-Agent Systems.* Revista Iberoamericana de Inteligencia Artificial, n°13, pp. 68-84, AEPIA. 2001.
16. Katia P. Sykara: *Multiagent systems.* AI Magazine 19(2). 1998.

MARCS Multi-agent Railway Control System

Hugo Proença[1] and Eugénio Oliveira[2]

[1] Universidade da Beira Interior, IT, Inst. Telecom.,
UBI, R. Marquês D'Ávila e Bolama, 6200-001, Covilhã, Portugal
hugomcp@di.ubi.pt
[2] FEUP/LIACC-NIADR,
FEUP, Av. Dr. Roberto Frias 4200-465, Porto, Portugal
eco@fe.up.pt

Abstract. Previous research works have demonstrated that traffic control models based on the comparison between an historical archive of information and current traffic conditions tend to produce better results, usually by improving the system's proactivity behavior. Based on this assumption, we present in this paper MARCS - Multi-Agent Railway Control System, a multi-agent system for communications based trains traffic control. For this purpose we have developed a system infrastructure based on an architecture composed of two independent layers: "Control" and "Learning".

"Control" layer is responsible for traffic supervision, regulation, security and fluidity, including three distinct agent types: "Supervisor", "Train" and "Station".

The "Learning" layer, using situations accumulated by the "Control" layer, will infer rules that can improve traffic control processes, minimizing waiting time and stop orders sent for each train. At this moment, inferred rules seem like:

"At T_1 moment, when a train is located at $P_1 = (x1, y1)$ with destination E_1 and another one is at $P2 = (x2, y2)$ with destination E_2, a traffic conflict in L_1 after t_1 seconds" will occur.

Rules of this kind are transmitted to the control system to be taken into account whenever a similar traffic situation is to occur. In the learning process we apply an unsupervised learning algorithm (APRIORI).

1 Introduction

1.1 Motivation

Railroad traffic volume will have in the next two decades a significant increment. More people and merchandize will circulate in increasingly bigger and faster railway networks [4].

Although traffic's scheduling systems guarantee that, for the foreseen conditions, vehicles in circulation will not compete simultaneously for the same resources (do not conflict), they lack of flexibility in order to enforce security.

Once railway networks will be under dynamic conditions it is most desirable that the control system becomes more flexible knowing how to provide an answer to these new requirement in an autonomous way.

Our propose consists in a completely distributed, decentralized and adaptable architecture for railway traffic control in a communications based train control [3].

1.2 Summary

The proposed system can be decomposed in two different sub-systems: “Control” and “Learning”.

“Control” sub-system is responsible for traffic management and guidance in the network and includes three agent types: (“Supervisor”, “Train” and “Station”). Those agents must interact with the objective of providing control mechanisms to the displacement of each train in the network. In this context, security is the principal concern, trying to assure that train crashes will never occur. Having security guaranteed, it is then important to maximize systems’s overall efficiency, by minimizing conflicts between trains. In our terms, a trains “conflict” occurs when several trains wish to cross at same time the same place. When that situation occurs, control sub-system is responsible for assigning priority, and sending “stop” commands to trains that must wait until the resource crossing zone is free.

“Learning” sub-system is the complementary one and has the objective of analyzing system’s past accumulated situations descriptions and identify typical cases that became the origin of later conflicts. The objective is to make the learning sub-system to infer rules that anticipate train conflicts and be able to make the “Control” system to benefit from them.

“Control” sub-system must compare all current train positions with the ones that can been identified by each rule. If any match is found, the predicted conflict must be avoided and all necessary actions must be taken in order to do it.

As we will show in section 4 this proactive behavior tends to minimize train conflicts and, under certain conditions, improve system’s efficiency.

1.3 Related Work

Probably induced by increasing traffic congestion problems, multi-agent systems applied to transportation domain, usually concern road transportation.

Typical approach tend to divide covered area by several traffic management agents, each one coping with decision responsibilities in a specific parcel. Often, authors decide to implement agents that represent every vehicle in circulation and the self physical infrastructure.

“TraMas” [7] is a system aiming at the study of multi-agent systems viability in the road transportation domain, as well as testing applicability of different models and cooperation strategies in multi-agent systems. In TraMas every cross point is represented by a traffic agent that is responsible for respective traffic control. Each one of the traffic control agents is independent enough to decide locally, but is also prepared to share information (cooperate) with other agents.

TraMas system includes three layers (Cooperative, Decision and Control).

[8] proposes a peculiar road traffic multi-agent system. Main objective consists in vehicles movement coordination inside a delimited area. Having a GPS-like localization method, every vehicle is represented by an agent, and each network parcel is managed by a control agent. Each control agent is responsible for analyzing specific traffic

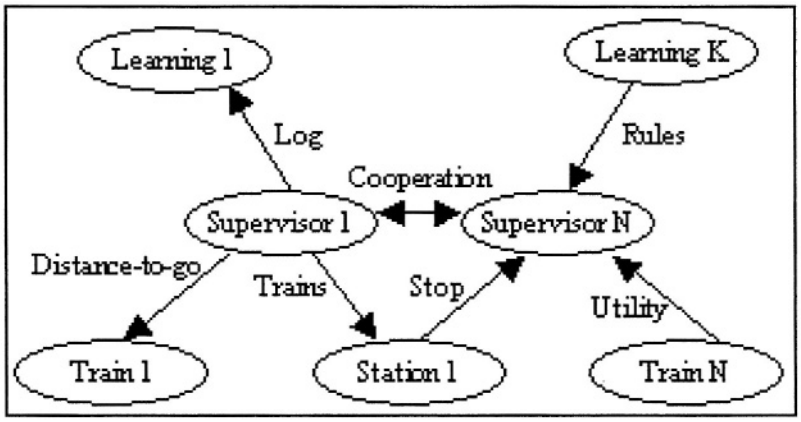


Fig. 1. MARCS architecture

volume and construct the system essential's component: "co-field". The co-field is a 3D representation of the environment, having two principal characteristics:

- Areas with high traffic density are represented as higher altitude (mountains).
- Areas with low traffic density are represented as lower altitude (depressions).

Vehicle agents are responsible for route selection through minimization travel cost. As expected, it is cheaper to travel in descendent directions. This fact induces vehicles to avoid areas with high traffic density, potentially where they could spent more time in result of traffic conflicts.

Proposed in 1994, dMARS [9] is a multi-agent system where two agent types interact: "Intersection" and "Street". They establish cooperation mechanisms with neighbors in order to produce an emergent system behavior. Every agent gets as input the traffic volume in represented area, being responsible for maintain that information and provide it to neighbors agents that request it.

[12] propose a multi-agent system for trains traffic coordination with one peculiar characteristic: natural language interface. When a traffic agent fells a high uncertain degree about what the correct action is, it starts user interaction section. User will analyze current traffic situation and communicate correspondent action in an oral form. This information must be achieved by the agents, and applied in next similar situation.

Several multi-agent systems [11] [10] and models [13] have been proposed, each of them with specific characteristics but sharing with the ones referred above: area (and responsibilities) sharing and inclusion of cooperation mechanisms enabling to go from a local to global perspective.

2 MARCS Architecture

Figure 1 gives a global perspective of MARCS architecture, existing agents in both sub-systems ("Control" and "Learning"), and interaction mechanisms between them.

Following what has been said before, the figure shows the interaction between four distinct agent types:

- *Supervisor.* These agents must control, guide and guarantee security in traffic network for each specific area. Each area is delimited by latitude and longitude coordinates. *Supervisor* agents are the only ones that simultaneously belong to control and learning sub-systems.
- *Train.* This agent type exclusively belongs to control sub-system. Train agents represent correspondent interests and are responsible for train velocity control, depending on free-distances ("distance-to-go") assigned by *Supervisor* agents.
- *Station.* Represents the interests of a railway station, and also belongs exclusively to control sub-system. Station agents objectives consist in administrate platforms, giving orders for trains arrivals and departures and providing useful users (passengers in train stations) information.
- *Learning.* This agent type belong to "Learning" sub-system. Learning agents task consists in asking for control agent registry log, analyze it, to identify possible existent meaningful patterns to infer possible rules that can optimize traffic fluidity.

2.1 Control Sub-system

Control sub-system main objective consists of providing secure and efficient routing for all trains while preventing crash situations and maximizing traffic fluidity.

This is accomplished through *Supervisor, Train* and *Station* agents information exchange consisting of both data and plans.

2.2 Learning Sub-system

As it was reported above , MARCS learning sub-system includes two different agent types: *Learning* and *Supervisor,* being this one also part of control sub-system. Performance and efficiency were primary factors analyzed at design time, inducing these architecture based on two parallel sub-systems.

Usually control systems have rigid time requirements making them adequate for "real time". On other hand, learning processes (specially data mining ones) tend to consume much computational resources and spend too much time until useful results become available. For our application, it was crucial that control processes priority was preserved, as well as guaranteeing that each real time traffic situation could be effectively analyzed, with no interference of any other task or objective that an agent possibly could have.

Another important factor is system modularity, which could also facilitate overall implementation. Once learning process tasks are easily distinguished from control ones, it becomes more intuitive the implementation by means of different computational entities.

Based on these requirements, *Learning* agents are learning sub-system essential components. Their unique objective consists in asking registry activity to *Supervisor* agents, concatenate and analyze it and infer rules that potentially increment system' s efficiency.

At creation time, each *Learning* agent receives a list of *Supervisor* addresses and becomes responsible for periodically asking for their activities record.

This consists in a “log” file maintained by each agent. It contains all received and sent messages and most relevant actions taken. For example:

```
1065189498 SEND     tell:sender supervisor1:receiver train1:content Distance 290.474074
1065189501 RECEIVE  tell:sender simulator1:receiver supervisor1:content Location train1 112.2 21.3
1065189501 RECEIVE  tell:sender simulator1:receiver supervisor1:content Location train2 213.2 -125.9
1065189501 PROCESS  Reserve vertex 12 Train train1
1065189501 SEND     tell:sender supervisor1:receiver train1:content Distance 230.882501
1065189501 SEND     tell:sender supervisor1:receiver train2:content Distance 75.127011
1065189503 RECEIVE  tell:sender simulator1:receiver supervisor1:content Location train1 117.2 55.1
1065189503 RECEIVE  tell:sender simulator1:receiver supervisor1:content Location train2 158.4 1.4
1065189503 PROCESS  Conflict Vertex 13 Trains 2 train1 train2
```

After complete log file content’s transmission, *Supervisor* work in learning process's compass is complete, being all following tasks performed by *Learning* agents, like related in the next sections.

3 Learning Process

Proposed learning process consists in the analysis of potential conflict situations (Section 2.1) that can occur and identification of train positions that originated those conflicts. If similar traffic conditions will repeat, control system must anticipate that conflict and take necessary actions to prevent and avoid it.

Figure 2 shows learning process state diagram. It consists of a preliminary phase of data pre-processing, followed by the algorithm execution, result analysis and knowledge acquisition. In a final phase, new knowledge is sent back to control sub-system, hoping that it will contribute for traffic fluidity and system effectiveness, by avoiding traffic conflicts.

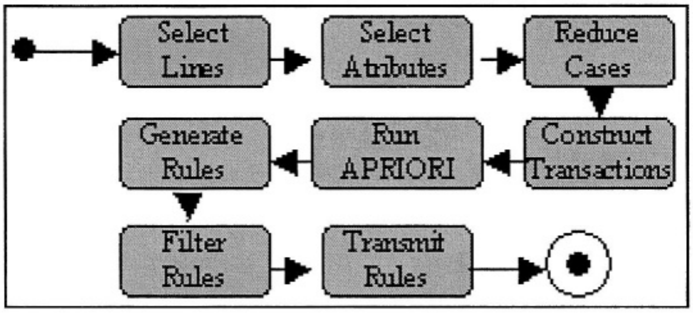


Fig. 2. Learning Process (State Diagram)

3.1 Data Pre-processing

An agent activity record consists in a text file containing all exchanged messages plus relevant actions taken.

Every line has format:

<time> <type> <description>

Where *type* can be one of "send", "receive", "process" or "exception" respectively for sent and received messages, actions taken or exceptions handled. *description* contains message or action additional description.

In the pre-processing phase there are four stages [5]: line selection, attribute selection, instance reduction and transaction construction.

In the end, data is formatted in an adequate way to be dealt with by next learning phase.

Transaction Construction. The next step consists in building transactions needed for algorithm execution. In the APRIORI [2] context terms, a transaction consists of an items set grouped by any criteria. For this purpose, we will group items relative to historic train locations that later originate a traffic conflict. A conflict identification between two trains (C_a and C_b) in location L_a and moment T_t, implies the selection of lines relative to absolute position of C_a and C_b in past $(t - i)$ moments, $i = 1..n$, being all grouped in the same transaction.

Let i_{conf} be a "Conflict" at time t_{conf} and C=$\{C_1, C_2, \ldots, C_n\}$ the set of trains involved. Define $\alpha \in N$, as the analysis retrospective limit. For each i_j line relative to C_j train location at t_j time: if $(t_j < t_{conf})$, $(t_j >= (t_{conf}$ - $\alpha))$ and $C_j \in C$ then add i_j to same transaction as i_{conf}.

As an example, see the set of items displayed below:

ID	Time	Type	Description	Destination
1	t_1	Location	Train C_1 (x_3, y_3)	Destination D_1
2	t_2	Location	Train C_2 (x_2, y_2)	Destination D_1
3	t_2	Location	Train C_1 (x_1, y_1)	Destination D_1
4	t_3	Conflict L_1	Trains C_1, C_2	
5	t_8	Location	Train C_3 (x_2, y_2)	Destination D_1
6	t_8	Location	Train C_4 (x_1, y_1)	Destination D_1
7	t_9	Conflict L_1	Trains C_3, C_4	

This set originates two transactions (T_1 and T_2):

Transaction	Items
T_1	1, 2, 3, 4
T_2	5,6,7

3.2 Algorithm for Association Rules

APRIORI allows the identification of association rules from large data sets grouped through transactions. Just as described in [2], let $\mathcal{I}$=$\{i_1, i_2, \ldots, i_m\}$ be a set of literals, called items. Let $\mathcal{D}$ be a set of transactions, where each transaction is an items set $\{i_j\}$, j=1..k, such that $i_j \in \mathcal{I}$. An *association rule* is an implication of form X $\Rightarrow$ Y, where X $\subset$ $\mathcal{I}$, Y $\subset$ $\mathcal{I}$ and X $\cap$ Y = $\emptyset$. The rule X $\Rightarrow$ Y holds in the transaction set $\mathcal{D}$ with *confidence* c if $c\%$ of transactions in $\mathcal{D}$ that contain X also contain Y. This rule has support s if $s\%$ of transactions in $\mathcal{D}$ contain X $\cup$ Y.

For a transactions set $\mathcal{D}$, the problem of mining association rules consists in generate all association rules with support and confidence higher than a specific threshold.

The logic of this algorithm is based on the observation that if any given set of attributes S has support lower than the threshold, any superset of S will have lower support and consequently any effort to calculate the support for such supersets will be wasted. For example, if we know that {A,B} is not supported it follows that {A,B,C} or {A,B,D} will also not be supported [6].

3.3 Rule Generation

After the execution of APRIORI [2], we have identified a frequent sets group $\mathcal{F}$, such that $\mathcal{F}=\{F_1, F_2, \ldots, F_n\}$, being each F_i an items set. Let $F_i=\{i_1, i_2, \ldots, i_n\}$ be a frequent set with dimension n. Adding a constraint to force that consequent only have one item, we build a group of n association rules R, with $R=\{\forall \text{ i} \in F_i : (F_i - \text{i}) \Rightarrow \text{i}\}$.

For our propose, we consider relevant association rules those with consequent type equal to "Conflict", meaning that we are interested in identifying those items (states) that usually occur together with a "Conflict" item.

For every frequent set F_i with dimension n, we can identify an association rule set R_i of dimension m (m < n), such that:

$$r \in R : \{i_{r1}, i_{r2}, ..., i_{rk}\} \Rightarrow i_{conf}, Type(i_{conf}) = "Conflict"$$

Like referred above, filtering only two distinct items type ("Location" and "Conflict"), all identified rules during the learning process have the form: **"IF** Train C_1 is at (x_1,y_1) with destination D_1 **AND** Train C_2 is at (x_2,y_2) with destination D_2 **AND** ... **THEN** Conflict in L_1, Trains $(C_1, C_2, \ldots)$, Time t.

Rule Selection. After rules generation process, it is important to select the most relevant ones and communicate them to the control system's agents. Above described process allows the identification of a large association rules set, most of them irrelevant. For instance, an inferred rule that will foresee a conflict between trains in the next second is not of great utility. On the other hand, if two rules r_1 and r_2 foresee conflicts for the same trains at the same place in, respectively, t_1 and t_2 seconds $(t_1 < t_2)$ with the same support and confidence, then r_1 is irrelevant, because there is another rule that anticipates the same situation at a former stage.

For the rules selection process we have defined a lower threshold l_i ($l_i > 0$) and consider every rule $r_i=(i_1, i_2, \ldots, i_n) \Rightarrow i_{conf}$, with Time$(i_{conf}) < l_i$ irrelevant. In a second phase we compare every remaining rule with each other, to analyze if exists a better rule for same situation.

Consider r_i a rule that anticipate a conflict at L_i place in about t_i seconds. Also let $(i_{l1}, \ldots, i_{ln})$ be locations of most ancient states attached to n trains involved in L_i conflict. If exists a rule r_j that also anticipate a conflict at L_i in t_j seconds $(t_j < t_i)$ with $(j_{l1}, \ldots, j_{ln})$ as respective locations for trains and there is only a single possible path between respective j_{lx} and i_{lx} places then r_i is irrelevant.

3.4 Rule Transmission

After the rule selection process, a relevant set of rules has been identified and we can proceed for last stage, its communication to control sub-system agents. This process concerns about trains localization, *Supervisor* agents identification, those with responsibilities for traffic management at the specific place, and rules transmission.

Having a rule r_i, such that $r_i=(i_1, \ldots, i_n) \Rightarrow i_{conf}$, we analyze parameter "Position" of every item i_j, j=1,..n, and determine who's *Supervisor* is responsible for those coordinates. Hereinafter, we proceed for rules transmission in KQML coded messages like the one exemplified in section 4.

This message informs $Supervisor_1$ that, if two trains have positions of respectively (861.0, -4.9) and (714.2,-263.1) and travel with direction $Station_1$ and $Station_2$, it will occur a traffic conflict in about 61 seconds, at Vertex 0, being Vertex 0 the internal representation of a specific railway cross point (switch point).

3.5 Control System Repercussions

On analyzing present train positions, *Supervisor* agents make the comparison with every received rule and, finding a match, proceed to "conflict avoid" process.

At this moment, this process consist in asking evolved trains to increase or decrease their usual velocity by $\alpha\%$ during t seconds, α,t > 0, conforming with the conflict time foresaw.

Traveling during a time interval at superior or inferior velocity then desired, can be enough for trains to arrive at predicted point at different moments avoiding the conflict and improving system's performance.

4 Experimental Results

Evaluation of a traffic control system could be done under multiple perspectives, perhaps with subjective components about the selection of most relevant characteristics.

"Security" should always be on top of priorities. It is crucial to assure that the system provide sufficient security mechanisms to avoid train collisions. "Capacity" and "Efficiency" could also be system evaluation parameters.

In our traffic simulator system, we implemented five evaluation parameters:

Crashes Number of train collisions.
Average Velocity Average velocity trains.
STOP Number of stop orders sent for trains.
Time Proportion at Desired Velocity Every train has a optimal velocity, according to its physic characteristics. This parameter represents the time proportion ([0,1]) that trains traveled at a velocity near the optimal one.
Simulation Time Total time spent until all trains arrive at final stations.

Experiments on several simulation scenarios like displayed on figure 3, allow us to conclude that rules inferred by the learning system have improved system performance by reducing "Stop" and "Simulation Time" parameters, and increase "Average Velocity" and "Time Proportion at Desired Velocity" without compromising security:

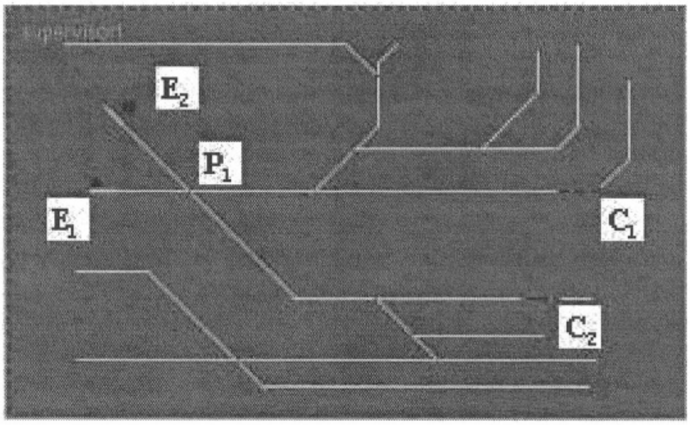


Fig. 3. Traffic simulation process

Parameter	**Before Learning Process**	**After Learning Process**
Crashes (Total)	0	0
Stop (Total)	1	0
Average Velocity (Km/h)	61	66
Time Proportion at Des. Vel. [0,1]	0.366	0.282
Simulation Time (mm:ss)	1.54	1.48

In the scenario displayed in figure 3 learning sub-system has identified the rule:

```
(tell:sender Aprender1
     :receiver Supervisor1
     :content (RULE
        Confidence 1.0
        Time -67
        TotalPremisses 2
            Premisse Local 861.0 -4.9 Destination Station2
            Premisse Local 768.0 -260.0 Destination Station1
        Consequent
            Conflict Vertex 0)
)
```

Rule displayed above informs “Supervisor” that if two trains with position (861.0, -4.9) and (768.0, -260.0) move respectively to "$Station_1$" and "$Station_2$" it will occur one traffic conflict at vertex $0\,(P_1)$.

Repeating simulation process, we observe that "$Supervisor_1$" demand "C_2" to reduce his optimal velocity, being this action enough to avoid the conflict at P_1.

5 Conclusions and Work in Progress

Experimental results allow us to conclude that, in specific cases, MARCS performance has been improved by applying learning system’s inferred rules.

Extending the learning process perspective, it is our intention to apply it to actions performed by the system in result of conflict anticipations. By now, learning system has the ability to anticipate conflicts, but cannot select the best action to avoid it, and cannot also anticipate whether actions performed to avoid a specific conflict will induce other future conflicts herder to be resolved.

Our work is currently focused on the analysis of the effects of actions performed to avoid conflicts, and determine those which are the optimal ones.

For this purpose, we plan to improve "line selection" phase, passing to select "Action" instances too. These elements specify actions taken by *Supervisor* agents with the aim of avoiding a conflict.

Having this, we expect to infer new knowledge represented in the following example: "Having a train C_1 located at P_1=(x_1, y_1) with destination D_1 and another C_2 located in P_2=(x_2, y_2) with destination D_2 it will occur a conflict in L_1 in t_1 seconds. **The best way to avoid this conflict is ask C_1 to decrease 10% his average velocity. Train C_2 must not accelerate because it will conflict with another one (C_3) in L_2 about t_2 seconds later ($t_2 > t_1$)**".

Conflicts transitivity is other factor that we also plan to analyze, grouping conflicts that apply to common trains. We want to derive optimal actions to avoid groups of conflicts and not just isolated ones.

References

1. A.G. Hobeika, C. F. Kim: Traffic flow prediction systems based on upstream traffic. Vehicle navigation and information systems IEEE conference (1994).
2. R. Agrawal, A. Strikant: Fast algorithms for mining association rules. VLDB (1994).
3. Transportation Systems Design: Communications Based Train Control. http://www.tsd.org/cbtc (2003).
4. Toby Moncaster: More Room On The Tracks. The future of railway signalling. http://www.essex.ac.uk/ese/siddiquiaward/ (2002)
5. Pavel Brazdil: Data Mining. Master Course in Artificial Inteligence and Computation. Universidade do Porto (2002).
6. Frans Coenen: The Apriori algorithm. Department of Computer Science. University of Liverpool. http://www.csc.liv.ac.uk/ frans/Notes/KDD/AssocRuleMine/apriori.html (2001)
7. Eugénio Oliveira and José M. Fernandes: TraMas - Traffic Control Through Behaviour Based Multi-Agent System. http://www.ieeta.pt/jfernan/tramas/ (1999).
8. Marco Mamei and Michael Mahan: Engineering Mobility in Large Multi-Agent Systems. SELMAS 2002 (110–122) (2003)
9. Tihomir Gabri and Emma Norling and Gil Tidhar and Liz Sonenberg and Nicholas Howden: Multi-agent Design of a Traffic-Flow Control System. (1994)
10. Vikram Manikonda and Renato Levy and Goutam Satapathy and David Lovell and Peter Chang and Anna Teittinen: Autonomous Agents for Traffic Simulation and Control. Transportation Research Record 1774 (1–10) (2003)
11. Danko A. Roozemond and Jan L.H. Rogier: Agent controlled traffic lights. ESIT(2000)
12. Alexander Huber and Bernd Ludwig: A Natural Language Multi-Agent System for Controlling Model Trains. http://www-wv.informatik.uni-erlangen.de/ bdludwig/pubs/huber_ludwig_camready.pdf (2002)
13. M. Antoniotti and A. Göllü: SHIFT and SMART-AHS: A Language for Hybrid System Engineering, Modeling and Simulation. USENIX Conference on Domain Specific Languages, Santa Barbara, CA (1997)

Dynamic Quality Control Based on Fuzzy Agents for Multipoint Videoconferencing*

Jesús Bobadilla[1] and Luis Mengual[2]

[1] DIA, Computer Science, U.P.M., Crta. de Valencia, Km 7,
28031 Madrid, Spain +34 913365055
jbobi@eui.upm.es

[2] DLSIS, Computer Science, U.P.M., Boadilla del Monte,
28660 Madrid, Spain +34 913367397
lmengual@fi.upm.es

Abstract. Real-time multimedia streaming applications are becoming increasingly popular on the Internet; however, the users of these streaming services often find their quality to be insufficient. When the services are based on multipoint videoconferencing, it is difficult to reach a constant, suitable level of quality on the video transmissions. In this paper, we propose to increase the overall quality of multipoint videoconferencing by dynamically acting on the JPEG quality parameter of each individual videoconference. To do this, we assign a fuzzy agent controller to each single videoconference. Since the agent's fuzzy logic is based on frame ratios (and JPEG qualities), the videoconference qualities will be dynamically equilibrated when a bottleneck is reached in any of the hardware resources supporting the system. This work includes a complete set of tests where the results of the fuzzy agents are compared with the optimum values reached by each multipoint videoconference: the proposed fuzzy architecture provides very good dynamic control of the videoconference qualities; moreover, its use will be particularly interesting in mobile environments, where the devices are heterogeneous and present limited processing capabilities.

1 Introduction

Multimedia streaming applications are becoming increasingly popular on the Internet; however, the users of these streaming services often find their quality to be insufficient. To improve this situation, there is currently an increasing amount of research on the different types of congestion control on the networks [1]. These controls can be implemented using different artificial intelligence mechanisms, such as fuzzy logic [2, 3] or agent systems [4], and it is common to combine these mechanisms [5].

Videoconferencing services are becoming more available on the Internet and their uses cover a large range of areas [6, 7]. Videoconferencing specifically requires a set

* This work is partially supported by the *Spanish Ministry for Science and Technology* within the following projects: *"A Multi-Agent System Architecture for the automatic and dynamic Implementation of security protocols (TIC2001-3376)"* and *"Open System of Multimedia Digital Presentation in Real Time (TIC2000-1734-C03-02)"*.

of resources that are not always available over time [8], and this situation can lead to the above-mentioned congestion, which needs to be controlled [9].

The quality control of video transmissions allows different alternatives to traditional networking approaches. One of these alternatives is to operate on the application layer, usually controlling the video coders [10, 11]; this way, we can aim to transmit slow video signals over the public network in real-time.

If a single videoconference communication can cause important congestion problems, then multipoint videoconferencing requires careful control of the resources in general and the bandwidth in particular [12]. This is a situation where it is advisable to act on the application layer to minimize congestion situations on the network layer. This paper focuses on this subject.

Rate control mechanisms are currently being used in multimedia streaming [13, 14], and some of them act on the application layer and some others introduce adaptation methods for UDP traffic.

This publication describes the design of a fuzzy-based agent controller which dynamically changes the JPEG quality of its associated videoconferencing. The paper does not describe the implementation details: the system has been implemented using the Sun Microsystems Java Media Framework API, which provides streaming, capturing, processing and rendering facilities, as well as suitable RTP access.

The paper starts with the architecture of the system and continues with its design; the architecture section focuses on the agent characteristics of the controllers: each controller acts on a single videoconference and implicitly shares the frame ratio variations with the rest of the agents.

The design section includes a study of the main factors of the application level that affect the videoconferences and the advisability of incorporating them in the fuzzy controller. The kernel of the design section presents the linguistic variables and the fuzzy rules.

The publication includes a complete section of results: real test results and simulated test results. The simulated results make it possible to compare the quality of multipoint videoconferencing in two situations: the optimum situation and the situation reached using the fuzzy controllers.

2 Fuzzy Agent-Based Videoconferencing System

The fuzzy agent we propose can be applied to any videoconferencing system where there are any resources shared by two or more videoconferences. Usually, the most critical shared resource will be the communications network, but it is possible to share different resources, such as computers acting as senders or receivers of multiple simultaneous videoconferences (Figure 1).

Each fuzzy agent acts on a different videoconferencing process; therefore, a system supporting '*n*' simultaneous videoconferences will form a fuzzy multi-agent consisting of '*n*' fuzzy agents. Periodically, each fuzzy agent receives its corresponding videoconference frame ratio, and dynamically it decides the JPEG quality that should be applied to improve the video quality (in order to maximize the frame ratio and JPEG quality).

Figure 1 shows the system architecture; senders use a communications network to deliver video frames using RTP, while receivers take the video streams and communicate the frame ratio to their corresponding agents. The fuzzy agents use the frame ratio to dynamically determine the JPEG quality it will be necessary to apply.

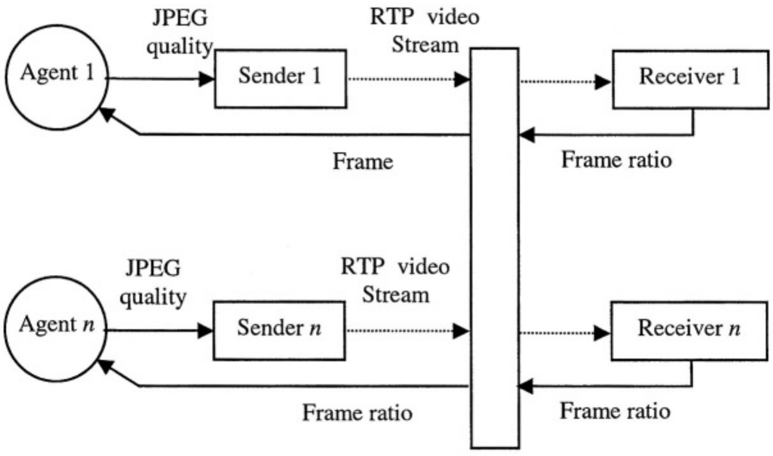


Fig. 1. Fuzzy multi-agent videoconferencing architecture

The fuzzy multi-agent videoconferencing system created can work using limited hardware resources and it has been tested for controlling several simultaneous videoconferences running on different computers. This means that it is possible to use each single computer to support several agents and senders. In this case, some frame ratios can be slowed down due to the excess load applied to the computers, and the fuzzy agents will determine the new JPEG qualities accordingly. The agents aim to maximize the videoconference qualities independently of the resources (network, computers, coders, effects, etc.) that produce the congestion.

We must realize that the different agents of the multi-agent system **implicitly** share some important information: the frame ratio variations. When there is any type of bottleneck in the system, the frame ratio of the videoconferences involved in the bottleneck drops, and, therefore, the fuzzy agents will try to compensate for this, probably by reducing the JPEG quality of their videoconferences. This action contributes to reducing the consequences of the bottleneck and to equilibrating the quality of the different videoconferences.

To explain the JPEG quality/frame ratio relationship, we carried out a test using four different videoconferences (160x120 to 640x480 resolutions) on a specific hardware configuration (AMD 2.4 GHz, Logitech camera, Sun Microsystems coder). Figure 2 shows the results. As you can see, low JPEG qualities determine high frame ratios, and high JPEG qualities (specially over 0.8) determine low frame ratios. This is due to the compressed frames achieved using low JPEG qualities.

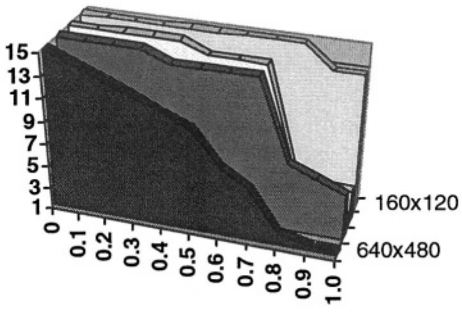


Fig. 2. JPEG quality (x-axis) versus frame ratio (y-axis) using different resolutions (z-axis)

It would be possible to design more sophisticated architecture, where the fuzzy-agents could be parameterized in order to take advantage of the specific configurations (number of senders and receivers on a computer, CPU power, bandwidth of the network, device and coder used, resolution of each videoconference, etc.) and each specific process' distribution, but the result would be multi-agent architecture that is difficult to tune to each specific configuration. Furthermore, 'harmonizing' these physical parameters would require some type of centralization or complicated distributed communications.

The solution we propose in this paper provides very efficient results based on a simple design of the fuzzy multi-agents; this simple design comes from the fact that several videoconferences sharing common resources will slow their frame ratios in any bottleneck situation. Consequently, their corresponding fuzzy-agents will independently reduce the JPEG quality (and therefore the overall bit-rate) as if it were acting in a more complicated distributed way.

3 Fuzzy Agent Design

It is necessary to study the impact of the different factors that affect the videoconferencing quality in order to design a suitable fuzzy agent that works properly on the possible different situations and environments. This will be done in the next section (3.1). When the above-mentioned factors have been considered, we can study different design possibilities, discarding or including some of these factors in the final designed approach. The fuzzy solution requires the selected factors to be converted into linguistic variables and fuzzy rules to be established that will act on the linguistic variables in order to provide satisfactory results (section 3.3).

3.1 Preliminary Considerations

Since we have determined that the frame ratio will feed the fuzzy agents, it would be advisable to study the different factors that can affect it (and that can be controlled by the fuzzy agents). Of course, we know the impact of the JPEG quality (Figure 2), but there are some other factors:

Coders: Each implementation of a coder can be more or less efficient; and more importantly, the actual nature of the coder determines a video compression capability and a bit-rate and frame-rate result.

Effects: Each frame of a videoconference can be processed to modify the video data, often to create effects, such as border enhancement, filters, etc. Depending on the relationship between the CPU power, the effect complexity and the resolution, the frame rate can drop if the computer cannot manage the effect in real time.

CPU Power, Resolution, Operating System, etc.: By carrying out an in-depth study of the factors that affect the frame ratio of one video transmission, we will find a combination of the basic factors we have outlined.

3.2 Proposed Agent Parameters and Technology

From the previous section, we can determine that it is possible to improve the videoconferencing frame rates by acting on different factors: JPEG quality, video resolution, selected coder, CPU power, operating system, etc. Some of the factors, such as the CPU power or the operating system, can clearly only be applied before starting each videoconference, since any attempt to dynamically change the factor would interrupt the transmission.

The video resolution and the selected coder could be changed dynamically, but this action would interrupt the current RTP video streaming, forcing a new one to be started. This would produce a long interruption that cannot be tolerated in the short periods the fuzzy agents should be acting to control the overall video quality.

Acting on the processing load of the effects is an interesting possibility for specific videoconferencing systems where effects are present and they can be parameterized. For example, we could dynamically change the level of an algorithm that enhances fuzzy images depending on the frame rate that the computer running the effect is able to obtain (this computer will usually be simultaneously running other time-variant load processes such as capturers and coders). This paper will avoid controlling any specific effect in order to offer a general solution to the videoconferencing dynamic quality control issue.

We select the JPG quality factor as the single parameter we will change to balance and enhance the overall videoconferencing quality. Figure 2 shows its importance in achieving this goal.

From Figure 2, we can see that by increasing the JPEG quality factor (JPG) we could decrease the frame ratio (FPS) and vice versa; this would lead us to work not only with the absolute frame ratio and JPEG quality factors, but also with their corresponding differential ones:

$$JPG_{t+1} = \eta\ (\Delta FPS_t,\ \Delta JPG_t),\ \text{where}\ \Delta FPS_t = FPS_t\text{-}FPS_{t-1}\ \ \&\ \ \Delta JPG_t = JPG_t\text{-}JPG_{t-1}$$

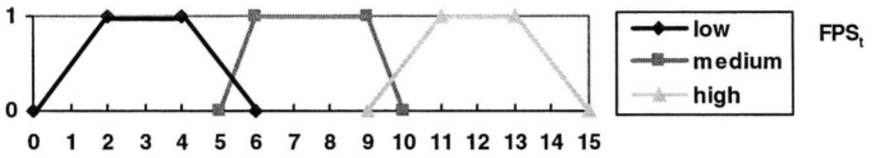
Since 'η' is not linear and it is not easy to determine, we must use an empirical method to implement it or a more sophisticated artificial intelligence approach. The predicated based logic could help to establish the behaviour of 'η', due to the natural way of expressing general rules, such as:

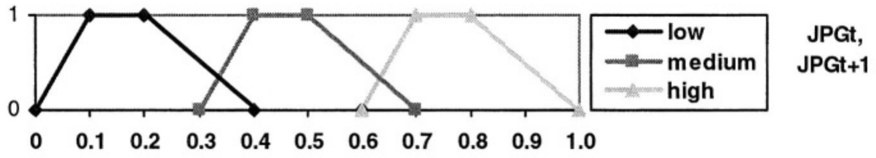
if ΔFPS_t is negative AND JPG_t is high then JPG_{t+1} is medium

Finally, we considered the fuzzy logic mathematical tool to be the most appropriated, due to the possibility to implement fuzzy behaviours and the difficulty to establish specific values for the limits of the predicates.

3.3 Fuzzy Agent Details

By adopting the decisions we made in the design sections, we are able to obtain an extremely simple and efficient fuzzy controller. Its linguistic variables are: FPSt, **ΔFPS,** JPGt, ΔJPG_t and JPG_{t+1}.





ΔFPS_t has two categories: negative (from -15 to 0) and positive (from 0 to 15). Similarly, ΔJPG_t also has negative (from -1 to 0) and positive (from 0 to 1) categories.

The fuzzy rules are divided into two main groups: static control rules and dynamic control rules. The first group looks after the behavioural changes that can be established by responding to the current state of the system, without looking at the recent changes the videoconferencing has experimented. That is to say, the rules with the $JPG_{t+1} = \eta\ (FPS_t, JPG_t)$ pattern. The dynamic control rules respond to the parameter changes: $JPG_{t+1} = \eta\ (\Delta FPS_t, \Delta JPG_t, JPG_t)$, reacting to the recent changes experimented in the videoconference parameters:

Table 1. Fuzzy rules

Static control rules
if FPSt is low and JPGt is high then JPGtp1 is medium
if FPSt is low and JPGt is not high then JPGtp1 is low
if FPSt is medium and JPGt is high then JPGtp1 is medium
if FPSt is medium and JPGt is not high then JPGtp1 is low
if FPSt is high and JPGt is high then JPGtp1 is high
if FPSt is high and JPGt is medium then JPGtp1 is medium
if FPSt is high and JPGt is low then JPGtp1 is low

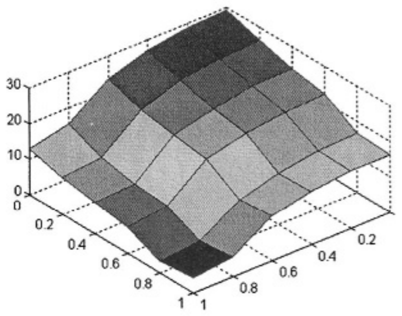
Table 1. (*continued*)

Dynamic control rules
Entering and exiting videoconferencing cases
if I_FPS is positive and I_JPG is positive and JPGt is low then JPGtp1 is medium
if I_FPS is positive and I_JPG is positive and JPGt is not low then JPGtp1 is high
if I_FPS is negative and I_JPG is negative and JPGt is not high then JPGtp1 is low
ifI_FPS is negative and I_JPG is negative and JPGt is high then JPGtp1 is medium
stabilised cases rules
if I_FPS is positive and I_JPG is negative and JPGt is high then JPGtp1 is medium
if I_FPS is positive and I_JPG is negative and JPGt is not high then JPGtp1 is low
if I_FPS is negative and I_JPG is positive and JPGt is not high then JPGtp1 is low
if I_FPS is negative and I_JPG is positive and JPGt is high then JPGtp1 is medium

4 Results

We have run a set of real multipoint videoconferencing processes to test the fuzzy controllers. Figure 3 (left) shows the average frame ratio reached for two simultaneous videoconferences using different JPEG qualities: (x & z axes). Figure 3 (right) shows the videoconferencing qualities (*Q*) obtained, where $Q_t = 1/25*FPS_t+JPG_t/5$. *FPSt* varies from 0 to 15 and *JPGt* varies from 0 to 1. The *FPS* parameter has 3 times more importance than the *JPEG* one. *Q* varies from 0 to 0.8.

The arrows inside Figure 3 (right) represent the 3 iterations the fuzzy controller executes to reach its maximum quality result.



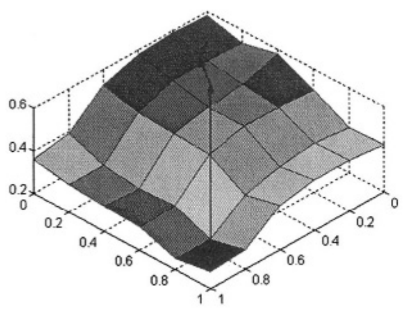


Fig. 3. Frame ratio (left) and quality (right) applying different JPEG qualities (x & z axes)

In order to test the fuzzy agent's behaviour exhaustively, we have used a network/system simulator configured to determine the different frame ratios obtained by applying 3 simultaneous videoconferences. We must provide the simulator with the following parameters: bandwidth of the network, resolutions and JPEG qualities of the 3 videoconferences and the MHz of the 3 computers we use to send the RTP video streams. The CPU powers have been set to 2000, 2400 and 3000 for the test we will explain in this section. The maximum frame ratio values have been set to 15; this can be considered as a good balance for multipoint videoconferencing systems.

Figure 4 (left) shows the frame ratios calculated for the simulator, providing different Kbps bandwidths (x-axis) and using three videoconferences of 160x120 (low), 320x240 (medium) and 640x480 (high). Figure 4 (right) shows the frame ratio obtained by varying the JPG quality (x-axis) and the bandwidth (z-axis); resolution 320x240.

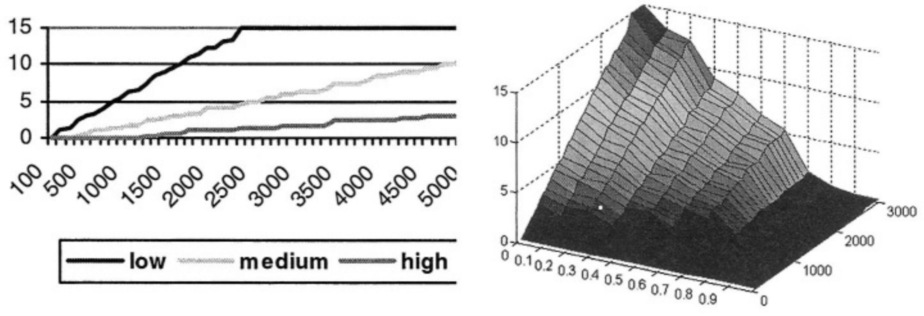


Fig. 4. Frame ratios generated for the system simulator

Using the simulator we have designed a first test where we calculate a good approximation of the average maximum video quality $\{Q_t = 1/25*FPS_t+JPG_t/5,\ Q \in (0..0.8)\}$ reached by applying 3 different resolutions (three 160x120, three 320x240 and three 640x480 videoconferences). For this purpose, for each bandwidth value, the simulation processes the 11^3 combinations it is possible to obtain by applying JPEG qualities 0 to 1 in steps of 0.1. We will call the results "Simula_Max". Next, we run the fuzzy agent for each bandwidth to calculate the quality it is able to obtain. Our aim is to get all the results close enough to the calculated maximums (the optimum values). This would mean that our fuzzy agent is dynamically choosing the adequate JPEG qualities; we will call the results obtained "Fuzzy_Average". Figure 5 shows the test results, as you can see, the fuzzy agent presented works well.

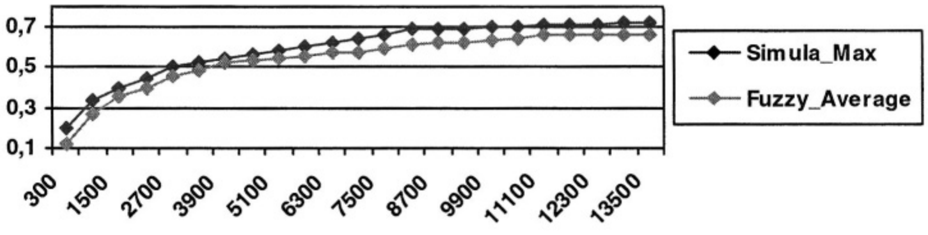


Fig. 5. Most favourable qualities and their corresponding fuzzy agent results (x-axis: bandwidth, y-axis: video quality $Q \in [0..0.8]$)

The last test included in this work (Figure 6) calculates "Simula_Max" and "Fuzzy_Average" for all the cases obtained by combining seven different sets of videoconference resolutions and the 33 bandwidth values ranging from 300 Kbps to 20000 Kbps. Figure 6 (left) shows the "Simula_Max" optimum values; Figure 6 (right) shows the fuzzy agent results "Fuzzy_Average". It can be seen that the fuzzy agent provides a very good approximation to the most favourable results. The flat area on the right figure can be improved by increasing the number of categories in the *JPG's* linguistic variables.

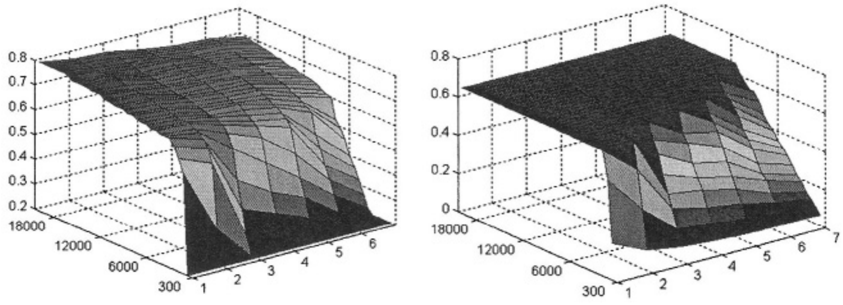


Fig. 6. Most favourable qualities (left) and their corresponding fuzzy agent results (right). x-axis: bandwidth, z- axis: resolution (1 means three 160x120 simultaneous videoconferences & 7 means three 640x480 simultaneous videoconferences)

5 Conclusions

It is possible to significantly increase the overall quality of multipoint videoconferencing by dynamically acting on the JPEG quality parameter of each individual videoconference. To do this, we can assign a fuzzy agent controller to each single videoconference. Since the fuzzy logic of the agents is based on the frame ratios (and JPEG qualities) the videoconference qualities will dynamically equilibrate when a bottleneck is reached in any of the hardware resources supporting the system.

The fuzzy agents are simple and efficient and they do not need to init or tune hardware-dependent parameters. The test results show that the JPEG qualities obtained by the fuzzy rules are very close to the optimum values, and these results can even be improved by incorporating new categories into the JPEG linguistic variables.

The proposed architecture runs on the application layer, and therefore, it is compatible with the typical networking congestion level algorithms; furthermore, its use will be particularly interesting in mobile environments, where the devices are heterogeneous and present limited processing capabilities. This idea will be the focus of our future work, combined with the dynamic control of real-time specific parameterized effects and their impact according to the CPU power of the multipoint videoconferencing senders.

References

1. Byunghun S., Kwangsue C., Yongtae S.: SRTP: TCP-Friendly Congestion Control for Multimedia Streaming, Information Networking. Wireless Communications Technologies and Network Applications (ICOIN), (2003) 529-539
2. Chrysostomou C., Pitsillides A., Rossides L., Sekercioglu A.: Fuzzy logic controlled RED: congestion control in TCP/IP differentiated services networks, Soft Computing - A Fusion of Foundations, Methodologies and Applications, Vol. 8, Issue 2, (2003) 79-92
3. Kazemian H., Meng L.: A fuzzy approach to video transmission over bluetooth ACL links, Consumer Communications and Networking Conference (CCNC), (2004) 533-538
4. Hashimoto K., Shibata Y., Shiratori N.: Mobile Agent-Based Adaptive Multimedia Communication, Conference on Information Networking. (ICOIN), (2002) 467-477
5. Delgado M. et al: A communication model based on the 2-tuple fuzzy linguistic representation for a distributed intelligent agent system on Internet, Soft Computing - A Fusion of Foundations, Methodologies and Applications, Vol. 6, Issue 5, (2002) 320-328
6. Anderson R., Beavers J., VanDeGrift T., Videon F.: Videoconferencing and presentation support for synchronous distance learning, Frontiers in Education (FIE), Vol. 2, (2003) 13-18
7. Tian J., Tianfield H.: A Multi-agent Approach to the Design of an E-medicine System, Lecture Notes in Computer Science, Vol. 2831, (2004) 85-94
8. Suganuma T., Imai S., Kinoshita T., Sugawara K., Shiratori N.: A flexible videoconference system based on multiagent framework, IEEE Transactions on Systems, Man and Cybernetics, Vol. 33, Issue 5, (2003) 633-641
9. Li Z., Xiao L., Ce Z., Yang X., Feng G.: A congestion control strategy for multipoint videoconferencing, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 12, Issue 11, (2002) 1025-1029
10. Lancini R., Fumagalli M., Arsura, E.: Scalable and robust videoconference codec: International Symposium on Video/Image Processing and Multimedia Communications (2002) 377-382
11. Garcia J., Brunstrom A.: A Robust JPEG Coder for a Partially Reliable Transport Service, International Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services (IDMS), (2000) 14-22
12. Sheikh H., Liu Z., Wang Z., Bovik A.: Foveated multipoint videoconferencing at low bit rates, IEEE International Conference on Acoustics, Speech, and Signal Processing, Vol. 2, (2002) 2069-2072
13. Yoshimura T., Ohya T., Kawahara T., Etoh M.: Rate and robustness control with RTP monitoring agent for mobile multimedia streaming, IEEE International Conference on Communications, Vol. 4, (2002) 2513-2517
14. Ahmed T., Mehaoua A., Boutaba R., Iraqi Y.: IP Video Streaming with Fine-Grained TCP-Friendly Rate Adaptation, Lecture Notes in Computer Science, Vol. 2839, (2003) 312- 322

A Component and Aspect-Based Architecture for Rapid Software Agent Development*

Mercedes Amor, Lidia Fuentes, and José María Troya

Dpto. Lenguajes y Ciencias de la Computación,
Universidad de Málaga, Campus de Teatinos,
s/n 29071 Málaga, Spain
(pinilla, lff, troya}@lcc.uma.es

Abstract. Current agent architectures provided by MAS platforms impose some limitation that affect the development of the functionality of software agents from scratch, placing little emphasis on (re)configuration and (re)use. This paper presents a software agent architecture development approach using a component and aspect–based architecture that promotes building agents from reusable software components and the configuration of some software agents. The basis of our architecture is the use of component-based and aspect-based software development concepts to separate agent functionality into independent entities increasing extensibility, maintainability and adaptability of the agent to new environments and demands. The architecture simplifies the software agent development process, which can be reduced to the description of the agents' constituent components and supported agent interaction protocols using XML documents. In addition, the extensibility provided by the component orientation enables to extend and reconfigure the internal agent architecture to accomplish additional agent capabilities such as planning.

1 Introduction

To accomplish a massive use of the agent technology depends, among others, on improving the developing and the deploying of multi-agent systems using existing platforms. Constructing software agents using any of the most accepted agent platforms can be a complex and error prone task that requires from the developer some skills in a concrete programming language, usually an object-oriented one. As a matter of fact, developers must became experts in using an API, provided by a platform vendor, so normally they are not interested in spending more time in learning how to use other agent platforms. Our aim is to facilitate developer's work, reducing the programming task and promoting the (re)use of agents inside multiple platforms.

Current agent architectures are mainly implemented as object-oriented frameworks [1,2,3] that provide a collection of extensible classes modeling typical

* This research was funded in part by the Spanish MCYT under grant TIC: 2002-04309-C02-02.

agent concepts. Component-Based Software Engineering (CBSE)[4] is a product of the natural evolution of object-orientation, endowing object-oriented languages with assembly capabilities beyond inheritance. While objects are expressed on the language level, components are expressed principally by exploring their public interface and promoting black box reuse [5]. We propose a component-based architecture for developing software agents that decompose agent functionality into independent components, i.e. in-house or COTS (Commercial Off-The-Shelf) components, or Web services [7].

Several benefits derive from developing software agents using a component-based architecture. For instance, it enables the construction of software agents mainly by composing the agent from reusable COTS components, reducing the development times, costs and efforts. Using highly tested COTS components would be a great accomplishment for agents, since it can save hours of development and the resulting agents should work better. Moreover, resulting component-based agents are more adaptive and flexible, as the agent functionality is coded independently from other agent specific classes.

However, decomposing a system, and in particular agents, in context independent components is not a trivial task. Commonly, the same concern happens to be spread over different components creating undesirable dependencies among them. Applying the separation of concerns principle, these concerns should be identified and separated to cope with complexity. According to what AOSD (Aspect Oriented Software Development) proposes [6] these concerns are modelled as first order software entities. Our agent architecture separates functionality from coordination and distribution aspects modelling them as different and decoupled entities. In this way the agent functionality is provided by reusable software components, which offer agent core services as well as application-dependent functionality, and are inserted in the agent as plug-ins. The composition between agent internal components is performed at runtime, allowing the reconfiguration and adaptation of agent behaviour to support new interaction protocols and functionality.

The coordination issue is separated in an independent entity, decoupling agent functionality from the agent interaction it is involved. Likewise, platform-dependent features, such as the distribution of messages using the FIPA-compliant message transport service is performed by a distribution aspect, and the encoding of messages in a concrete FIPA representation is also enclosed in a different entity. Since platform dependencies are encapsulated as external plug-ins, our agents can be adapted to engage in any platform. Therefore, instead of defining just another agent platform we propose: "write your agent once, run it on any FIPA-compliant platform".

Another important contribution of our approach is that an agent can be "programmed" simply by editing XML documents describing agent constituent components. The developer only has to provide an explicit description of the supported interaction protocols, a description of the components that will be assembled inside the agent architecture, and other information related with MAS deployment.

The structure of this paper is organized as follows: Section 2 presents our component-based agent architecture, describing the main entities of our model. In section 3 we illustrate through an example how to extend the agent architecture to incorporate a planner that will be used to plan agent actions, and how to program an agent using this feature. Finally some concluding remarks are given in section 4.

2 The Component and Aspect-Based Agent Architecture

The design of any agent architecture can be afforded from a software engineering perspective viewing an agent as a complex piece of software that is involved in multiple concurrent tasks and requires interactions with other agents. Our architecture models key agent concepts such as functionality, coordination, and agent communication representation and distribution as separated entities (Fig. 1). Currently, this architecture is implemented in Java.

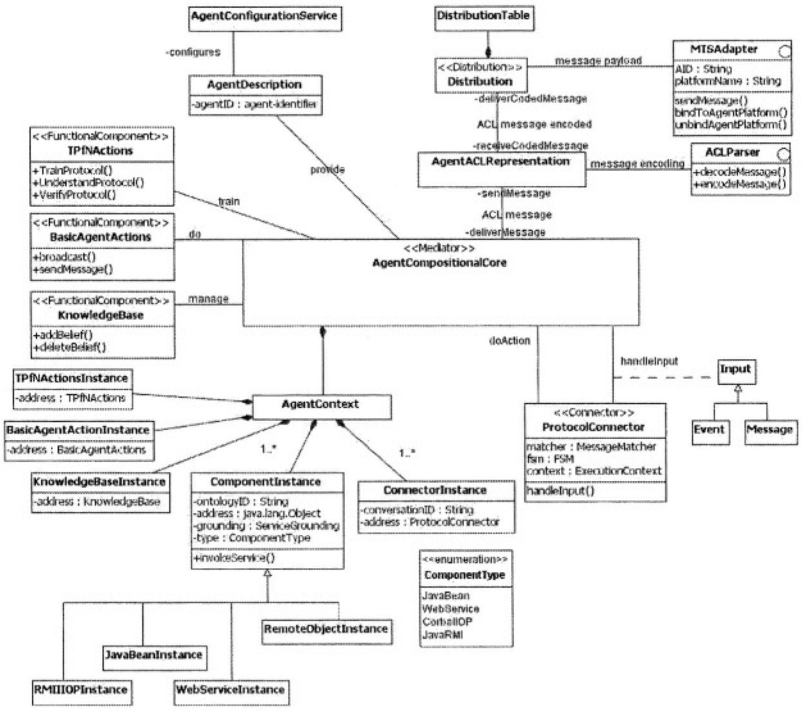


Fig. 1. Component and Aspect-Based Architecture for Software Agents

Agent Interface

The *AgentDescription* component contains the agent's public interface, which is an extension of the traditional software component's public interface adapted to software agents. In our case, the agent offers a public interface that includes a description of the agent's functionality, a list of communication protocols supported by the agent, the supported ACL formats of incoming and outgoing messages, and a list of agent platform names where the agent may engage. To unify the description of software components public interfaces avoiding the use of proprietary interface description languages (IDLs), we propose the use of OWL-S (before known as DAML-S). OWL-S consists of a set of ontologies that provide a vocabulary to describe services. The OWL-S profile and model descriptions provide enough information to enable automated execution based on well-defined descriptions of a service's inputs and outputs. The use of ontologies allows sharing a semantic agreement about the services provided by and included on an agent, and not just a syntactic one as traditional IDLs.

Functionality

In our architecture components labelled as <<FunctionalComponent>> (see Fig. 1) encapsulate data and functionality, and are always present in the architecture providing the general-purpose agent functionality. The basic functionality that every agent possesses, such as locally store and access shared data and the ability of constructing and sending a message, are provided in *KnowledgeBase* and *BasicAgentActions* components. An additional feature, but considered as basic in our architecture, to train the agent in a new protocol [8] is supported by the *TpfNActions* component. On the other hand, application and domain specific functionality can be provided by any software component, i.e. a COTS component or a Web service [7], by means of its offered services, which are accessed by public interfaces [5]. While in current agent architectures the agent behaviour is encapsulated in units, such as tasks or actions, and its implementation must adhere to a set of inheritances and interfaces realizations using a traditional object-oriented API, our model design does not impose any restrictions to software components added to the agent. These components are only required to describe their provided interface in OWL-S. These components may be plugged into the agent on demand, and are easily changeable over the lifetime of the agent, guaranteeing software agent evolution.

Coordination

In our architecture, an independent entity denoted as <<Connector>> controls the conversations in curse in which the agent is involved, and coordinates the execution of agent functionality that is invoked along a conversation. Otherwise we do not propose a centralized coordination engine to control agent interactions, but rather an independent connector deals with each agent interaction (see *ProtocolConnector* in Fig. 1). Each connector coordinates a dialogue according to a specific protocol (e.g. English Auction Protocol), which is not hard coded inside. The coordination protocol is described in XML and it is given to the connector at instantiation. We provide a XML Schema that must be followed by any specific protocol description in order to be properly processed by a connector. The protocol description does not only contain the rules to coordinate a conversation, but also it describes a detail plan for conducting a role that states the set of agent actions that are executed. These agent actions might refer to any service included in the agent interface and provided by a software component.

The description of an interaction protocol consists of the description of the interchanged messages and the description of the roles participating in the interaction. A separate finite state machine, represented by a set of state transition rules, depicts each role. Each transition description includes also the actions that the agent carries out when receiving a message or an internal event in the context of the interaction, but expressed in an implementation independent way by means of OWL-S.

At runtime, an agent is able to encompass a new protocol only by uploading the corresponding XML description through the TPfN (Training Protocol for Negotiation) protocol [8] defined for that purpose. Hence we provide a single connector implementation that can cope with the coordination of any protocol described following a XML schema that defines the information required to describe any protocol. This implementation is described in more detail in [13].

Dynamic Composition

The <<Mediator>> component *AgentCompositionalCore* (ACC) (showed in Fig. 1) performs the dynamic composition of components and connectors enabling loose coupling between them. The mediator maintains relevant information about components that provide agent functionality and active conversations controlled by connectors. This information is represented in the architecture by the object *AgentContext* (see Fig. 1), which holds references to all components and connectors instances, and uses the descriptions contained in the *AgentDescription* component to create connectors and components. The conversation identifier names active connectors *(ConnectorInstance),* and every component *(ComponentInstance)* is identified within the agent architecture with a kind of role name. Since the role a component plays inside an agent is determined by the ontology it commits, we use the ontology name for identifying components. *ComponentInstance* class has been subclassed to consider different types and features of plugged software components (Java Bean, Web Service, etc.).

Since components and connectors have no direct references among them, their composition is performed at runtime by the *ACC,* handling connector requests of performing agent functionality. Dynamic composition allows replacing a component instance by another implementation, upgrading the agent without substituting it. Moreover the agent configuration could change to meet a certain requirement such as efficiency. For example, an agent can test different Web services providers offering the same service, and use the one with the current best response time. In addition, this component is in charge of dispatching incoming messages and internal events to connectors. The conversation identifier of an incoming message is checked to determine if the message belongs to an active conversation or a new conversation.

Message Representation

The *AgentACLRepresentation* component allows the agent to communicate using different ACL representations. This component hides representation dependencies defining a high-level interface interface *ACLParser* (shown in Fig. 1) to send and receive messages in different ACL representations. A different parser supports the parsing and encoding of a message in a concrete ACL representation such as

XML, String or BitEfficient encoding formats following FIPA specifications. This component does not care about the source agent platform of the message, as ACL encoding is independent of the delivering platform.

Message Distribution

By separating the distribution of messages from the functionality and coordination it is possible to use different agent platform services for message delivery. In the architecture the distribution aspect hides platform dependencies defining a high-level interface to send and receive messages to and from different agent platforms. This separation allows adapting the agent to use the services of different agent platforms. It avoids to compromise agent development to platform dependencies, and reduces the management of communication [12]. In the architecture, the distribution aspect is encapsulated in the *Distribution component* (showed in Fig. 1). For every agent platform, this component instantiates a specialized component realizing the interface *MTSAdapter,* which encapsulates the corresponding platform-dependent functionality allowing the agent to engage different agent platforms by providing access to the corresponding Message Transport Service (MTS). Separating platform specific code in different components reduces the complexity of managing communication with multiple agent platforms. Since agent bootstrapping and MTS access can differ, every *MTSAdapter* implements platform specific mechanisms needed to send and receive legal messages from a concrete agent platform and use other platform services. The *Distribution* component is able to determine the appropriate adapter by using a distribution table with pairs (AID, platform identifier). Our implementation currently supports adaptors for FIPA-OS, JADE, and Zeus agent platforms.

3 The Development of a Planning Software Agent

This section explains how to build a planning agent using our approach. We have chosen this example to show that our agent architecture can be easily extended to cope with new agent capabilities, such as planning. In addition, this example will show how our architecture can afford the development of agents that do not include a fixed set of predefined plans defining their behavior. Instead, an inner planner generates the plan of actions of the agent at runtime. Currently, just a few FIPA-compliant general-purpose agent architectures, such as Zeus, allow explicitly the definition and use of plans to define agent behavior. In other well-known FTPA-compliant agent architectures, where agent behavior is modularized and separated into tasks, is quite difficult to afford the addition of new functionality that provides the agent with planning capabilities. We mean to the capability of generating a plan, that is, a sequence of agent actions. Actually, Jade agent architecture includes a scheduler, not a planner. Regarding FIPA-OS agent architecture, it defines a Planner Scheduler component that is not currently available. In our case, to extend our agent architecture with planning capabilities only entails the adding of an internal planner component, in charge of creating plans of actions that could achieve agent's goals.

The agent that will illustrate this example is a planning agent for the block world domain, a well-known domain in the planning research community. This agent owns a set of blocks, and its goal is to achieve a certain configuration of blocks. For this purpose, the agent is able of moving its blocks, but also it can request blocks to other agents in order to achieve the required configuration. In our model, agent functionality, such as moving blocks, is provided by services offered by plugged components and the interaction with other agents is controlled by means of FIPA-compliant protocols.

To develop and deploy an agent using our approach, we provide an XML deployment document (as the one depicted in Fig. 2) that the developer should fulfil with the initial configuration and properties of the agent. This information allows configuring the agent architecture. The description of the planning agent of our example will consist of a component that provides a service for generating plans, and another component that implements the typical actions of the block world domain, offered as well as services of its provided interface. These components that are initially plugged into the agent architecture are packaged into the *Functionality* element (see Fig. 2). Within this element, and for each plugged component, the element *Component* encloses information about its provided interface and its implementation. The component interface describes the set of offered services in an XML document in OWL-S, pointed in the *InterfaceDescription* element. The *DeploymentInfo* element points to a XML document containing information about the component implementation regarding the kind of implementation (e.g. Java, CORBA, Web service), how to locate and deploy the component, etc. As detailed in Fig. 2, the first component element includes the information bound to the *BWPlanner* component, that is, the XML document that describes, in OWL-S, the provided interface of the planner component, and the information for deploying the component. Likewise, the second *Component* element (see Fig. 2) details similar information relating to the second component, namely *BlockWorld.*

Within the *Coordination* element (see Fig. 2), the designer must specify the coordination protocols supported by the planning agent. As stated in the previous section, the agent coordination and the communication between agent inner components is decoupled from agent functionality and described in XML documents; therefore, this section of the agent description includes references to the XML protocol description documents. Each XML protocol description is referenced in the *href* attribute of a *ProtocolDescription* element. The coordination of our planning agent is given by three coordination protocols. The first *ProtocolDescription* element in Fig. 2 refers to *BWPlanning.xml,* which describes the coordination among the *BWPlanner* component, the *BlockWorld* component and the *BasicAgentActions* component (shown in Fig. 1). This protocol governs the initiation, negotiation and execution of a plan. In addition, the planning agent supports two different coordination protocols that involve interaction with other agents. The second *ProtocolDescription* element in Fig. 2 points to the description of the FIPA Contract-Net protocol for contracting the block supplier. The third *ProtocolDescription* element in Fig. 2 points to the document containing the XML description of the FIPA Request protocol for requesting a block from other agent.

```
<?xml version="1.0" encoding="UTF-8"?>
<AgentDescription xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="C:\map\tesis\xml\AgentDescriptionSchema.xsd">
 <Behaviour>
  <Functionality>
   <Component>
    <InterfaceDescription href="http://map:8081/idl/BWPlanner.owl" notation="OWL-S"/>
    <DeploymentInfo href="http://map:8081/idl/PlannerDeployment.xml" notation="String"/>
   </Component>
   <Component>
    <InterfaceDescription href="http://map:8081/idl/BlockWorld.owl" notation="OWL-S"/>
    <DeploymentInfo href="http://map:8081/idl/BlockWorldDeployment.xml" notation="String"/>
   </Component>
  </Functionality>
  <Coordination>
   <ProtocolDescription href="http://map:8081/protocol/BWPlanning.xml"/>
   <ProtocolDescription href="http://map:8081/protocol/BWFIPAContract_Net.xml"/>
   <ProtocolDescription href="http://map:8081/protocol/BWFIPARequest.xml"/>
  </Coordination>
 </Behaviour>
 <Distribution>
  <AgentPlaftorm adaptor="http://map:8081/adapters/fipaosAdapter.class" platformName="FIPA-OS"/>
 </Distribution>
 <ACLRepresentation ACLParser="http://www.altova.com/ACLString.class" format="String"/>
 <KnowledgeBaseContent>
   +<AcquaintanceDatabase resource="BlockSuppliers">
 </KnowledgeBaseContent>
 <ActiveContext>
  <StartProtocol protocolID="BWPlanning" ontologyID ="BlockWorld" role="planning">
   <hasInput>
    +<Input named="suppliers">
   </hasInput>
   <hasInput>
    +<Input named="goal">
   </hasInput>
  </StartProtocol>
 </ActiveContext>
</AgentDescription>
```

Fig. 2. XML deployment document of a planning agent

Within the *Distribution* element, the developer details the agent platform adaptors, including a reference to its implementation. The *AgentPlatform* element in Fig. 2 includes the adaptor for the FIPA-OS platform. The *ACLParser* plug-in formats ACL messages in the String format. This information is given inside the *ACLRepresentation* element in Fig. 2.

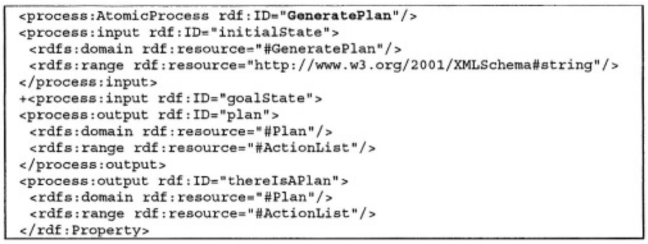
The description of the planning agent also includes, enclosed in the *KnowledgeBaseContent* element, the initial content of the agent Knowledge Base, expressed in terms of beliefs, goals, conditions, and acquaintance databases, that defines a set of the identifiers of the agents with which the agent will interact. As detailed in Fig. 2, the knowledge base component of the planning agent will contain an initial list of block supplier agents, defined as an acquaintance database named *BlockSuppliers*. Finally, within the *ActiveContext* element, the developer specifies the initial behaviour the agent will execute, expressed in OWL-S. The *ActiveContext* element of Fig. 2 encloses the start of the planning protocol.

The Planner Component

Since our agent architecture does not provide a specific component for planning, we need to implement a new component providing this capability. For this purpose we developed a software component wrapping a legacy planner. This planner, which follows the SNLP Algorithm, is implemented in Common Lisp, and tries to generate a plan to achieve a goal given an initial state. The planner uses an extended *block world* domain description that includes three predicates and one additional action. The predicates *(have x)* and *(not-have x)* state, respectively, that there is or there is not a block x on the table. The predicate *(available x)* states the there is a block x available in the universe. The action *(demand x)* procure a block x. This action allows introducing communication with other agent in order to acquire new blocks.

A wrapper component, named *BWPlanner,* offers a service for generating a plan, accepting as inputs the initial state and the goal state. As stated before, the provided interface of the wrapper component is described in OWL-S in *BWPlanner.owl.* This interface comprises a single service, named *GeneratePlan,* whose OWL-S description as an OWL-S atomic process is depicted in Fig. 3. The OWL-S description includes a description of the inputs required by the service (named *initialState* and *goalState* in Fig. 3), and the outputs generated (named *plan* and *thereIsAPlan* in Fig. 3), all of them defined as properties of the process *GeneratePlan.*

We want to point out that, since the planning capabilities are encapsulated in a software component, it is possible to use a different planner in the architecture providing another implementation, maintaining the same provided interface or extending it including a new service to generate more than one plan. For this purpose, the planner can be adapted to provide more than one plan, in one step or in successive requests.



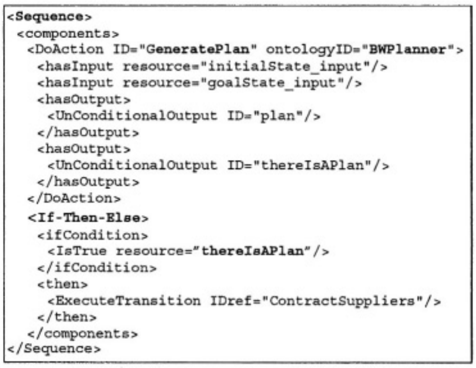
```
<process:AtomicProcess rdf:ID="GeneratePlan"/>
<process:input rdf:ID="initialState">
 <rdfs:domain rdf:resource="#GeneratePlan"/>
 <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</process:input>
+<process:input rdf:ID="goalState">
<process:output rdf:ID="plan">
 <rdfs:domain rdf:resource="#Plan"/>
 <rdfs:range rdf:resource="#ActionList"/>
</process:output>
<process:output rdf:ID="thereIsAPlan">
 <rdfs:domain rdf:resource="#Plan"/>
 <rdfs:range rdf:resource="#ActionList"/>
</rdf:Property>
```

Fig. 3. OWL-S description of the *GeneratePlan* service provided by the planner component

The Planning Protocol

In our agent model, the invocation of the agent functionality, provided as services by plugged components, is described in the transitions of the protocol description. In the case of our planning agent, the transitions description included in the *BWPlanning* protocol describes how to coordinate the invocation of the services provided by the *BWPlanner* component, the *BlockWorld* component and the *BasicAgentActions* internal component to generate a plan, negotiate contracts with block suppliers and execute every action of the plan.

Every *TransitionDescription* element encloses references to agent actions, such as generating a plan, moving blocks, and starting a new conversation with other agent requesting a block, that are invoked during the planning protocol execution. Within every transition description, and using control constructors defined in OWL-S, such as Sequence, or If-Then-Else, we can specify how to coordinate the execution of agent functionality. The Fig. 4 depicts the transition description that is executed at the beginning of the planning protocol. The *Sequence* control constructor encloses a list of processes to be invoked in a sequential order. As shown in Fig. 4, this transition consists in invoking the *GeneratePlan* service provided by the planner component to generate a plan. One of the components of the *Sequence* is an *If-Then-Else* control constructor, which allows putting a condition to the plan execution. If the output *thereIsAPlan* of the *GeneratedPlan* service (as detail in Fig. 3, the service output *thereIsAPlan* determines if the planner has found a plan), then the transition identified as *ContractSuppliers* is executed in order to start the block negotiation with the supplier agents.



```
<Sequence>
 <components>
  <DoAction ID="GeneratePlan" ontologyID="BWPlanner">
   <hasInput resource="initialState_input"/>
   <hasInput resource="goalState_input"/>
   <hasOutput>
    <UnConditionalOutput ID="plan"/>
   </hasOutput>
   <hasOutput>
    <UnConditionalOutput ID="thereIsAPlan"/>
   </hasOutput>
  </DoAction>
  <If-Then-Else>
   <ifCondition>
    <IsTrue resource="thereIsAPlan"/>
   </ifCondition>
   <then>
    <ExecuteTransition IDref="ContractSuppliers"/>
   </then>
  </components>
</Sequence>
```

Fig. 4. XML description of the initial transition of the BWPlanning.xml

4 Conclusions

The contribution of this work is a component and aspect-based agent architecture that combines component orientation and the separation of concerns principle. This proposal offers some benefits derived from the compositional approach: The (re)use of third-vendor components reduces programming errors during implementation, since they are supposed to be tested and error-free. Providing a COTS market, agents can be programmed simply by editing XML documents, reducing the time and effort required to develop agent-based applications. Since agent behavior is not hard-coded inside the agent, agent upgrade does not required to compile source code. Moreover, our agents benefit from agent platform independency due to the separation of the distribution concern applied, which lets them to live in most of the FIPA-compliant agent platforms.

In addition, though constituent agent components are particularized by the configuration and deployment information, new components can be plugged in at runtime to address new requirements, and already registered components can be updated with new releases. We do not impose any methodological approach for the design of the agent. Instead, we propose to apply MDA mechanisms to derive each agent implementation in our agent model, from a design model provided by an agent-oriented methodology [11].

References

[1] The Zeus Agent Building Toolki ,BtexaCT,. http://193.113.209.147/ projects/agents/zeus
[2] FIPA-OS, Emorphia. http://fipa-os.sourceforge.net/
[3] Java Agent Development Framework, TILAB, http://jade.cselt.it
[4] G.T. Heineman, W.T. Councill, "Component-Based Software Engineering: Putting the Pieces Together", Addison Wesley, 2001.
[5] C. Szyperski, Component Software: Beyond Object-Oriented Programming, Addison Wesley, 1998
[6] Aspect-Oriented Software Development. http://www.aosd.net
[7] M. Amor, L. Fuentes, J.M. Troya, "Putting Together Web Services and Compositional Software Agents", *Web Engineering,* LNCS 2722, 2003, pp. 44–53.
[8] M. Amor, L. Fuentes, J.M. Troya, "Training Compositional Agents in Negotiation Protocols" in *Integrated Computer-Aided Engineering International Journal.* Vol. 11, No. 2 pp. 179–194. 2004.
[9] Jess, The Java Expert System Shell, Distributed Computing Systems, Sandia National Laboratories, http://herzberg.ca.sandia.gov/jess/
[10] FIPA Interaction Protocols Specification, Foundation for Intelligent Physical Agents, 2000
[11] M. Amor, L. Fuentes, A. Vallecillo, "Bridging the Gap Between Agent-Oriented Design and Implementation", next publication in AOSE 2004.
[12] M. Amor, L.Fuentes, J.M. Troya, "A Component-Based Approach for Interoperability Across FIPA-Compliant Platforms", in *Cooperative Information Agents VII,* LNAI 2782, 2003. pp. 266–288.
[13] M. Amor, L.Fuentes, L. Mandow, J.M. Troya, "Building Software Agents from Software Components", in *Current Topics in Artificial Intelligence,* LNAI 3040, 2004. pp. 222–231.

Formalization of Cooperation in MAS: Towards a Generic Conceptual Model

Monia Loulou, Ahmed Hadj Kacem, and Mohamed Jmaiel

University of Sfax,
Laboratory LARIS-FSEGS,
B.P. 1088, 3018 Sfax, Tunisia
{Monia.Loulou, Ahmed}@fsegs.rnu.tn
Mohamed.Jmaiel@enis.rnu.tn

Abstract. This paper proposes a formal definition of a conceptual model for cooperation among multi-agent systems (MAS). This works constitutes a part of a general research project aiming at defining a generic interaction model that covers the different facets of the cooperative activity among MAS. We propose a framework for the specification, design and verification of interaction mechanisms. Doing so, and using the Z notation, we bring closer the concepts describing a cooperative activity while integrating them with the concepts related to the organizational modelling within a MAS. Moreover, we suggest a set of necessary properties needed to maintain the consistency at the individual level (Agent) as well as at the collective level (System). The syntax and the semantic of proposed specifications have been validated with the Z-eves verification tool [1].

1 Introduction

It is widely recognized that the specification phase in the design process of multi-agent systems is primordial, since it permits to master their complexity. Indeed, the use of formal specifications enables to avoid any ambiguity and imprecision and allows to reason rigorously about their properties and their behaviors. In general, the formal specification of multi-agent systems is an intricate task, due to the complexity of interaction mechanisms, the heterogeneity of system components and the lack of consensus about the fundamental concepts, such as agent, role, plan, objective, organizational structure, mental state, etc. In order to facilitate formal specification of multi-agent systems, we need a formal framework that clarifies these concepts, unifies their representation and defines the relations between them. Many formal frameworks for agent systems have been proposed, such as the model proposed by Luck and d'Inverno [2] using Z notation and the BDI model [3] using modal logic. However, most of these models are limited to particular domains, like [4] for distributed problem resolution. However, we notes the quasi absence of standardization works that defines a generic interaction model dedicated to all formal specification of a MAS regardless of the application domains. In the perspective to define a generic interaction model, that integrates the individual aspect (agent) as well as the collective aspect (society), we focus, in this study, on the modelling of the cooperation mechanisms. The method we adopted to define our generic model is based on the study of several multi-agent architectures applied in different application domains. In a preparatory stage, we draw up a list of concepts as well as the purposes they are used in different architectures. Then, we retain the concepts which appear to be appropriate to the concerned domain. Thereafter, we assign to each concept a unique purpose. This permits to unify them. In a second main step, we investigate the relations that exist between the retained concepts. This is based on their use in the different architectures. Finally, we make use of these architectures to formalize the cooperation mechanisms. The multi-agents systems are used in large application domains ; therefore it was necessary to restrict our survey to a limited range. In this work, we opted for some application domains relative to the category of systems formed by software entities that execute themselves in a virtual environment as : the multimedia applications domains ([5] and [6]), electronic commerce domains ([7] and [8]) and the Manufacturing systems ([9] and [10]). A study of other fields should be the subject of a future work to proof the generic aspect of the concepts which will be retained.

The retained concepts following the developed study on they selected architectures will presented in section 2. This section consists in defining the concepts allowing to qualify the cooperative aspect in a MAS. In the last section, the established models are instantiated on a multi-agent architecture in order to ensure its consistency. The chosen architecture is the MASPAR (Multi-Agent System for Parsing Arabic) [11]. We conclude this paper by summarizing our ideas and by describing the future directions of our research.

2 Formal Cooperation Model

This section treats the different aspects of cooperation in a MAS and the links between them in two phases : the cooperation model and the cooperative agent model. The first phase consists in defining the concepts permitting to qualify the activity in a cooperative MAS, while the second describes the internal aspects of a cooperative agent as well as its basic properties. This models are formalised using Z notation [12].

2.1 Cooperation Model

To define the cooperation model, we structure our presentation according to three levels : when can we talk about cooperation ? , how can this cooperation take place ? , how can we quantify the efficiency and the coherence of this cooperation ?. These points will be detailed in the following subsections.

When Can We Talk About Cooperation? According to the study of various multi-agent application domains, we noted that we can talk about cooperation if there is a society of agents that intend to reach a common objective. This society requires a set of roles organized under an organizational structure that determines the responsibility of each agent.

- A society of agents *"Society"* is composed of a finite set of agents, that collaborate to achieve a common objective, each one is defined as a simple agent or a subset of agents that inter-communicate and cooperate under an internal control.

  A mental state *"MentalState"* regroups the whole of individual agent properties. These laters are classified in three subsets: the capabilities *(Capability* $== \mathbb{P}$ *Propriety),* the knowledges *(Knowledge* $== \mathbb{P}$ *Propriety)* and the beliefs *(Belief* $== \mathbb{P}$ *Propriety).* A simple agent is defined by the *agent* schema.

$$\begin{array}{|l}
\hline \textit{MentalState} \\
\textit{capability} : \mathbb{P}_1\ \textit{Capability} \\
\textit{knowledge} : \mathbb{P}\ \textit{Knowledge} \\
\textit{belief} : \mathbb{P}\ \textit{Belief} \\
\hline
\end{array}
\qquad
\begin{array}{|l}
\hline \textit{agent} \\
\textit{MentalState} \\
\hline
\textit{belief} \cup \textit{knowledge} \neq \{\} \\
\hline
\end{array}$$

  Let $\textit{Agent} ::= A\langle\langle \textit{agent} \rangle\rangle \mid An\langle\langle \mathbb{F}\ \textit{agent} \rangle\rangle$.

$$\begin{array}{|l}
\hline \textit{Society} \\
\textit{agents} : \mathbb{P}_1\ \textit{Agent} \\
\hline
\#\textit{agents} \geq 2 \\
\hline
\end{array}$$

- A common objective *"CommonObjective"* is defined by a nonempty set of global goals *[GlobalGoal]* and a set of constraints *[Constraint]* related to the running modes of this objective. For abstraction reasons, we consider global goal as a set of local goals *[localgoal],* each one is described by a set of elementary tasks [*Task*]. Formally : $\textit{LocalGoal} == \mathbb{P}_1\ \textit{Task}$ and $\textit{GlobalGoal} == \mathbb{P}_1\ \textit{LocalGoal}$.

  The global goals of a same common objective are related by a variable dependence ratio such as : a total dependence *TD,* a partial dependence *PD* or an independence *ID.*

$$\begin{array}{|l}
\hline \textit{CommonObjective} \\
BG : \mathbb{P}_1\ \textit{GlobalGoal} \\
Cr : \mathbb{P}\ \textit{Constraint} \\
TD : \textit{GlobalGoal} \leftrightarrow \textit{GlobalGoal} \\
PD : \textit{GlobalGoal} \leftrightarrow \textit{GlobalGoal} \\
ID : \textit{GlobalGoal} \leftrightarrow \textit{GlobalGoal} \\
\hline
\forall Bg_1, Bg_2 : BG \bullet (Bg_1, Bg_2) \in TD \vee (Bg_1, Bg_2) \in PD \vee (Bg_1, Bg_2) \in ID \\
\hline
\end{array}$$

- A role *"Role"* is defined according to the agent's capabilities. In the studied architectures, we note that a same role can appear in various applications. Thus, we propose to define the role concept in terms of domain capabilities *[DomainCapability]* and of control capabilities *[ControlCapability].*

Formally : *DomainCapability* == $\mathbb{P}_1$ *Propriety* and *ControlCapability* == $\mathbb{P}_1$ *Propriety* as : *DomainCapability* ∩ *ControlCapability* = {} and *Domain Capability* ∪ *ControlCapability* = *Capability.*

$$\begin{array}{|l}
\textit{Role} \\ \hline
\textit{domainCapability} : \mathbb{P}\ \textit{DomainCapability} \\
\textit{controlCapability} : \mathbb{P}\ \textit{ControlCapability} \\ \hline
\textit{domainCapability} \cup \textit{controlCapability} \neq \{\} \\ \hline
\end{array}$$

Formally, the relation between capabilities and roles is defined by *AgentR* function.

$$\textit{AgentR} : \mathbb{P}_1\ \textit{Capability} \rightarrow \mathbb{P}\ \textit{Role}$$

– A generic organizational structure *OrgStructure* is defined by a finite set of relations *[OrgRelationship]* between the different roles required for the achievement of a common objective. The relations between a set of roles and a global goal are defined by the *SOrg* function.

$$\begin{array}{|l}
\textit{OrgStructure} \\ \hline
\textit{Oc} : \textit{CommonObjective} \\
R : \mathbb{P}_1\ \textit{Role} \\
\textit{Rorg} : \mathbb{P}_1\ \textit{OrgRelationship} \\
\textit{SOrg} : \textit{GlobalGoal} \times \mathbb{P}_1\ \textit{Role} \rightarrow \mathbb{P}_1\ \textit{OrgRelationship} \\ \hline
\#R \geq 2 \\
\forall\, Bg : \textit{Oc.BG} \bullet \forall\, r : \mathbb{P}_1\ \textit{Role} \mid r \subseteq R \bullet \textit{SOrg}(Bg, r) \subseteq \textit{Rorg} \\ \hline
\end{array}$$

The cooperation model is then described by the *CooperationModel* schema. This later verifies for each role, needed to reach *Oc,* that it exits at least one agent endowed with competence allowing him to take it in charge.

$$\begin{array}{|l}
\textit{CooperationModel} \\ \hline
S : \textit{Society} \\
\textit{Oc} : \textit{CommonObjective} \\
R : \mathbb{P}_1\ \textit{Role} \\
\textit{Sog} : \textit{OrgStructure} \\
\textit{Coop} : \textit{Society} \times \textit{CommonObjective} \times \mathbb{P}\ \textit{Role} \rightarrow \textit{OrgStructure} \\ \hline
\textit{Coop}(S, \textit{Oc}, R) = \textit{Sog} \\
\forall\, r : \mathbb{P}_1\ \textit{Role} \mid r \subseteq R \bullet \exists\, a : \textit{agent} \\
\quad \bullet \exists Ag : \textit{Agent} \mid A(a) = Ag \bullet Ag \in S.\textit{agents} \wedge r \subseteq \textit{AgentR}(a.\textit{capability}) \\ \hline
\end{array}$$

How Can This Cooperation Take Place? A cooperative activity can take place if particular set of agents converge towards a global goal when each one achieves his proper local goals. The local goals of an agent expresse the tasks assigned to it in order to perform a global goal. The assignment is based on the roles distribution, described by the *AgentGoal* function. This later verifies the completeness property related to the attribution of the local goals of a same global goal.

$$
\begin{array}{|l}
AgentGoal : CooperationModel \times GlobalGoal \times agent \rightarrowtail\!\!\!\!\!\!\!\rightarrow \mathbb{P}\, LocalGoal \\
\hline
\forall\, C : CooperationModel \bullet \forall\, Bg : GlobalGoal \mid Bg \in C.Oc.BG \\
\quad \bullet \forall\, a : agent \bullet \forall\, Ag : Agent \mid A(a) = Ag \wedge Ag \in C.S.agents \\
\qquad \bullet\, AgentGoal(C, Bg, a) \subseteq Bg
\end{array}
$$

The distribution of the roles induces an instantiation of the generic organizational structure, called concrete organizational structure *ConcreteOrg*. This structure can establish additional links needed for the resolution of the exceptions which can occur during a task progress. An organizational link *OrganizationLink* makesgrapt possible to associate two or several agents according to a type of links *[LinkType]* for a given ipplication context *LinkContext.*

For abstraction reasons, we consider *LinkContext* = *GlobalGoal.*

$$
\begin{array}{|l}
\textit{ConcreteOrg} \\
\hline
Sog : OrgStructure \\
S : Society \\
organizationlinks : \mathbb{P}_1\, OrganizationLink \\
\hline
\forall\, Or : organizationlinks \bullet \forall\, A : Agent \bullet A \in S.agents \Leftrightarrow A \in Or.Ag \\
\forall\, Or : Organizationlinks \bullet \forall\, l : LinkContext \\
\quad \bullet\, l \in Or.linkcontext \Rightarrow l \in Sog.Oc.BG \\
\hline
\end{array}
$$

How Can We Quantify the Efficiency and the Coherence of This Cooperation? In this section, we define two properties related to the quantification of the efficiency and the performance of a cooperative activity.

- Efficiency : to qualify the cooperative activity to be efficient, it's not sufficient to perform only local goals constituting the common objective. But, we suggest to satisfy the generic constraints to reach this objective. The *RespectConstraint* function measures the ability of a society of agents to achieve the objective according to the so-called constraints. Thus, for each agent, we verify according to the *RespectSequence* function, the existence of a sequence of tasks execution that respects the retained constraints. Let *bool* ::= *Yes* | *No*

  $$\begin{array}{|l} RespectSequence : \mathbb{P}\, Task \times \mathbb{P}_1\, Constraint \rightarrow bool \end{array}$$

  The *Mission* function assigns a mission to every agent that defines its part of responsibility in the society.

  $$\begin{array}{|l} Mission : Agent \times CommonObjective \rightarrow \mathbb{P}\, Task \end{array}$$

Thus, a *RespectConstraint* function is defined formally by :

$$
\begin{array}{|l}
RespectConstraint : Society \times CommonObjective \rightarrow bool \\
\hline
\forall S : Society;\ Oc : CommonObjective \\
\quad \bullet \forall a : Agent \mid a \in S.agents \\
\qquad \bullet \exists M : \mathbb{P}\, Task \mid M = Mission(a, Oc) \\
\qquad\quad \bullet RespectSequence(M, Oc.Cr) = Yes \Rightarrow \\
\qquad\qquad\quad RespectConstraint(S, Oc) = Yes
\end{array}
$$

- *Performance of the Organization* : it is about the verification for every agent the presence of a subset of Organizationnel links that allows him to acquire knowledge, through others, in order to achieve its task and avoid all deadlock situations. The function *ExecuteTask* determines the execution state of a task according to the organizational structure with which the agent is arranged. Let *State* ::= *lock* | *unlock.*

$$
\begin{array}{|l}
ExecuteTask : Agent \times ConcreteOrg \times Task \rightarrow \mathbb{P}\, Knowledge \rightarrow State \\
\hline
\forall A : Agent; Soc : ConcreteOrg \bullet A \in Soc.S.agents
\end{array}
$$

The verification of the performance of a concrete organization according to the achievement of a common objective is measured by the function *PerformOrg.*

$$
\begin{array}{|l}
PerformOrg : Society \times CommonObjective \times ConcreteOrg \rightarrow bool \\
\hline
\forall S : Society; Oc : CommonObjective; Soc : ConcreteOrg \\
\quad \bullet \forall a : S.agents; t : Task \mid t \in Mission(a, Oc) \\
\qquad \bullet \exists K : \mathbb{P}\, Knowledge \bullet ExecuteTask(a, Soc, t) = K \wedge \\
\qquad\quad ResultTask(a, t, K) = unlock \Rightarrow \\
\qquad\qquad PerformOrg(S, Oc, Soc) = Yes
\end{array}
$$

2.2 Cooperative Agent Model

In opposition to the cooperation model which treats the collective aspect, the cooperating agent model deals with the individual aspects of the agent. In order to obtain a unified and precise agent's definition, we try to capture the essence of an *(cooperating)* agent. Ideally, we attempt to produce some ingredients and properties for the cooperating agent which should be consistent with our cooperation model. The properties presented below are designed to characterize an agent and to adequately perform the cooperative activity.

We define a *cooperating agent* as an entity (software or hardware) living in an environment (may include other agents), which has capabilities and resources enabling it to perform individual tasks and to cooperate with other agents in order to reach a common objective. This common objective is reached by achieving agent's local goals. Starting from this informal definition we can associate to the previews description of the mental state, two representations : the accointances of the agent which represent its partial views on organizations to which it belongs and its agenda which describes the set of the agent's tasks to be performed on time. This agenda can be updated at any time by the agent itself.

$$\begin{array}{|l}
Agenda : Agent \rightarrow \mathbb{P}(Task \times Time) \\
\hline
\forall\, a : Agent \bullet \forall\, Oc : CommonObjective \bullet \\
\quad \forall\, t : Task \mid t \in Mission(a, Oc) \bullet \exists\, T : Time \bullet (t, T) \in Agenda(a)
\end{array}$$

So that the agent could progress evolute in a cooperative society, some properties must be considered to adequately perform the cooperative activity :

- *Autonomy* : An agent is said to be *autonomous* if it is able to make decisions for performing additional actions, or for changing its current task. Then, we suppose that an autonomous agent has the capability of choosing or changing his agenda at any time, according to its availabilities and its preferences.
- *Perception* : it is the capacity of an agent to perceive its environment and consequently to update its mental state.
- *Auto-Organization* : an agent is able to auto-organize himself, when it has the capability to evaluate his interactions with others and add organizational links or remove some of them.

Let *[EnvState]* consider a set of environment states. A cooperative agent is defined by a *Cooperative Agent* schema that verify the previously detailed properties.

$$\begin{array}{|l}
\hline
\textit{CooperativeAgent} \\
agent \\
agenda : \mathbb{P}(Task \times Time) \\
views : \mathbb{P}\ OrganizationLink \\
UpdateAgenda : \mathbb{P}(Task \times Time) \times EnvState \rightarrow \mathbb{P}(Task \times Time) \\
Perception : EnvState \times (Knowledge \cup Belief) \rightarrow (Knowledge \cup Belief) \\
AutoOrganization : EnvState \times \mathbb{P}\ OrganizationLink \rightarrow \mathbb{P}\ OrganizationLink \\
\hline
\forall\, Ag : Agent; a : agent \mid Ag = A(a) \bullet agenda = Agenda(Ag) \\
\forall\, a : agent \bullet \forall\, Oc : CommonObjective \\
\quad \bullet\ views = \{view : OrganizationLink \mid A(a) \in view.Ag\} \\
\hline
\end{array}$$

The described properties are necessary to lead convenably a cooperative activity. Also, we can extend this list to other properties such as the reactivity, the capacity of reasoning, the persistence, the intelligence, etc.

3 Illustration

We propose, in this section, the validation of the formal models, already defined, by their instantiation on the MASPAR system *(Multi-Agent System for Parsing ARabic)* for parsing arabic texts. MASPAR provides for each sentence of the analysed text the possible syntactic structures [11]. The agents implied in this system must satisfy, as presented in the *CommonObjective(MASPAR)* schema, four global goals : *Bg*_1 (text segmentation), *Bg*_2 (morphological analysis of the words), *Bg*_3 (syntactic analysis of the sentences) and *Bg*_4 (resolution of the complex linguistic forms such as the anaphoric forms and the ellipses). The resolution of this goals require five main roles. We denote this roles by:

R_Seg : split the text into sentences and segment each one in words,
R_Morph : correspond to the morphological analysis,
R_Syn : correspond to the syntactic analysis of the sentences,
R_Anph : resolve the anaphoric forms in the sentence,
R_Elip : detect and rebuild the various types of ellipses.

$CommonObjective(MASPAR)$
$BG == \{Bg\_1, Bg\_2, Bg\_3, Bg\_4\}$
$ID\ \ (Bg\_4, Bg\_1)$
$PD\ \ (Bg\_2, Bg\_4)$
$PD\ \ (Bg\_2, Bg\_1)$
$PD\ \ (Bg\_1, Bg\_3)$
$PD\ \ (Bg\_3, Bg\_4)$
$PD\ \ (Bg\_2, Bg\_3)$

As defined in our cooperation model, the roles must be organized according to an- organizational structure in order to collaborate in the achievement of a global goal.

In the MASPAR system, we identify three types of organizational links according to [13]: the accointance links(*Ro*1), the communicationel links *(Ro*2*)*, and the operative links *(Ro*3*)*. The set of the relations established between the different listed roles, constitutes the generic organizational structure of MASPAR so-called *OrgStructure(MASPAR)*. We present, as an example, the organizational links between the *R_Morph* role and the *R_Seg* role in order to achieve the *Bg*-1 global goal by: *SOrg(Bg_*1*, R_Morph, R_Seg)* = {*Ro*3*, Ro*2}.

The MASPAR architecture is based on a functional approach which associates to each agent one of the already cited roles. Thus, the whole of the agents is composed of: Segmentation Agent*(A_Seg),* Morphologic Agent *(A_Morph),* Syntactical Agent *(A_Syn),* Anaphoric Agent *(A_Anph)* and Elliptic Agent *(A_Elip).* The *A_Morph* is composed of two agents: an agent for the morphological pretreatments and another agent for the morphological tagging. This case illustrates the recursive aspect in the *Agent* definition.

The distribution of the roles among agents defines the concrete organizational structure which describes the organizational links between these agents. We present, as an example, the relations between *A_Seg* agent and *A_EMorph* agent by the *OrganisationLink(Seg_EMorph)* schema.

$OrganisationLink(Seg\_EMorph)$
$Ag == \{A\_Seg, A\_EMorph\}$
$linktype == \{Ro3\}$
$linkcontext == \{Bg\_1\}$

4 Conclusion

The generic modelling of the interaction mechanisms is still an open research domain. Indeed, this works proposes a generic conceptual cooperation model using the Z notation. This model defines the concepts describing a cooperative activity while integrating them with the concepts related to the organizational modelling. An instantiation of these concepts with MASPAR system [11] constitutes a first validation of the proposed models. Also, we proposed the specification of the necessary properties to maintain a consistency at the individual level (agent) as well as at the collective level (Society). This specification has been supported by the type checking and the theorem proving of the Z-Eves tools [1].

These results deserve some improvements. Indeed, it is recommended to associate the concepts related to coordination and negotiation with those of the cooperation in order to improve the action of the group either by an increase of the performances, or by a reduction of the conflicts. The association of concepts related to the cooperation, negotiation and coordination, constitutes our perspective for future works. This association permits to define a conceptual interaction model. In the second perspective we propose the definition of a cooperation and negotiation language like the coordination language proposed in [14]. This language will be defined by a precise syntax and an operational semantics in term of transition systems. These languages of interaction come to be integrated in a platform of design and implementation of MAS.

References

1. Meisels, I., Saaltink, M.: The Z/EVES 2.0 reference manual. Technical Report TR-99-5493-03e, ORA Canada, Canada (1999)
2. Luck, M., d'Inverno, M.: A formal framework for agency and autonomy. In: Proceedings of the first international conference on Multi-Agent Systems, AAAI Press/MIT Press (1995) 254–260
3. Rao, A.S., George, M.P.: Bdi agents: From theory to practice. In: In Proceedings of the First International Conference on Multi-Agent Systems ICMAS. (1995) 312–319
4. Jmaiel, M., Hadj-Kacem, A.: A formal definition of cooperation and agency among multi-agent systems. In: Proceedings of the International Conference on Artificial and Computational Intelligence for Decision, Control and Automation in Engineering and Industrial Applications, Monastir, Tunisia (2000)
5. Ford, B., Karmouch, A.: An architectural model for mobile agent-based multimedia applications. In: Proceedings of Canadian Conference on Broadband Research, Ottawa (1997)
6. Ziade, F., Karmouch, A.: A multimedia transportable agent system. In: Proceedings of IEEE Canadian Conference Electronic Computer Engineering, St. John's, NFLD (1997)
7. Tsvetovatyy, M., Gini, M., Mobasher, B., Wieckowski, Z.: Magma: An agent-based virtual market for electronic commerce. Journal of Applied Artificial Intelligence **11** (1997) 501–524

8. Oliveira, E., Rocha, A.: Agents advanced features for negotiation in electronic commerce and virtual organisations formation process. In: Agent Mediated Electronic Commerce: the European AgentLink Perspective. Number 1991 in LNAI, Springer Verlag (2000) 78–97
9. Shen, W., Xue, D., Norrie, D.: An agent-based manufacturing enterprise infrastructure for distributed integrated intelligent manufacturing systems. In: Proceedings of The Third International Conference on the Practical Application of Intelligent Agents and Multi-agents, London,UK (1998) 533–548
10. Shen, W., Barthes, J.: Dide: A multi-agent environment for engineering design. In: Proceeding of The First International Conference on Multi-Agent Systems, San Francisco, USA (1995) 344–351
11. Aloulou, C., Hadrich-Belguith, L., Hadj-Kacem, A., Ben-Hamadou, A.: Conception et développement du système MASPAR d'analyse de l'arabe selon une aproche agent. In: RFIA'04, 14ème congrès francophone de la Reconnaissance des Formes et Intelligence Artificielle, Toulouse, France (2004)
12. Bowen, J.: Formal Specification and Documentation using Z: A Case Study Approach. International Thomson Computer Press (1996)
13. Ferber, J.: Les systèmes multi-agents. Vers une intelligence collective. InterEdition (1995)
14. Jmaiel, M., Hadj-Kacem, A.: An operational semantics for negotiating agents. In: Intelligent Agent and Multi-Agents Systems. Volume 2413 of Lecture Notes in Artificial Intelligence., Springer (2002) 77–91

Web-Enabling MultiAgent Systems

Eduardo H. Ramírez and Ramón F. Brena

Centro de Sistemas Inteligentes,
Tecnológico de Monterrey
{eduardo.ramirez, rbrena}@itesm.mx

Abstract. In this paper we describe a component architecture for web-enabling MultiAgent Systems intended for the deployment of distributed artificial intelligence applications. This integration allows agents to publish web services or standard HTML providing thus a convenient interface for other distributed components. By using an Embedded Web Services approach we provide a simple and efficient mechanism for enabling agents to interoperate with its users and enterprise systems such as portals or client-server applications. The architecture we propose is presented from a structural as well as functional point of view. Comparisons are drawn with other proposals for integrating intelligent agents with web services, and experimental performance measurements show the advantages of our approach.

1 Introduction

In the past 15 years, agent technologies have evolved from its own set of standards, languages[1] and applications and it was not until the last years when the issues concerning interoperability between agents and more traditional applications began to be addressed. At the time of this writing we may say that agent development have found some stability at the implementation language level, having tools and frameworks in mature object-oriented development languages such as the JADE platform [2].

However, having software agents to interoperate with users and other applications using web-based interfaces remains as a critical issue for widespread adoption of agent technologies. In this paper we describe a component architecture for web-enabling MultiAgent Systems intended for the deployment of real-world distributed artificial intelligence applications. This integration allows agents to publish XML[3] Web Services[4] or standard HTML providing thus a convenient interface for other distributed components.

Throughout this discussion we will use the Web Service definition stating that a Web Service is “a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL [5]). Other systems interact with the Web Service in a manner prescribed by its description using SOAP-messages [6], typically conveyed using HTTP with an XML [3] serialization in conjunction with other Web-related standards.” [4].

We consider as well the following definitions:

– A *component,* according to Fielding[7], is "an abstract unit of software instructions and internal state that provides a transformation of data via its interface". Data transformed by a web component is intended to be used in HTTP conversations that may be HTML for web pages or XML [3] for Web Services. We consider the use of *Servlets* as they are the Java standard user-defined web components. JSR-154 defines them as "A Java technology-based web component managed by a container that generates dynamic content" [8].
– A *container* is "an entity that provides life cycle management, security, deployment, and runtime services to components" [9]. There exist well defined containers for software agents as well as for web components. Agent platforms like JADE[2] platform provide containers that function as execution environments for agents. As specified in JSR-154[8], web containers are usually built within web servers and provide network services related with HTTP request processing.

2 Architectural Approaches

As mentioned above, Web-enabling MultiAgent Systems requires interoperability both at component and at container level, however existing solutions address each concern separately. The main identified approaches are discussed as follows.

2.1 Component Interactions, a Gateway Approach

Authors of the JADE platform have defined a basic approach for communicating agents and web components, that conforms to the *transducer* approach identified by Nwana[10].

The solution assumes the existence of two agent containers, one standalone, that we may call the *main container* and one contained itself within the web container. Each container is executed in a separate JVM[1] system process. Whitestein Technologies A.G. developed the WSAI[11], an open-source product which refines and implements the initial architecture for use in the AgentCities[12] project.

WSAI introduces the concept of "Gateway Agent" as an agent living in the web container, responsible of translating HTTP requests into ACL messages. The general gateway architecture components are shown in figure 1.

One of the major drawbacks of the approach resides in the existence of several processes that should synchronize using remote method invocations even if both of them are deployed on the same machine. The synchronization problem is addressed by instantiating the "Gateway Agents" and "Caller" objects in a request basis. "Callers" implements the Java synchronization and time out handling as shown in figure 2.

An AgentCities[12] requirement that imposes several complexities to the WSAI, is that it was designed to work with any running FIPA-compliant agent

[1] Java™ Virtual Machine.

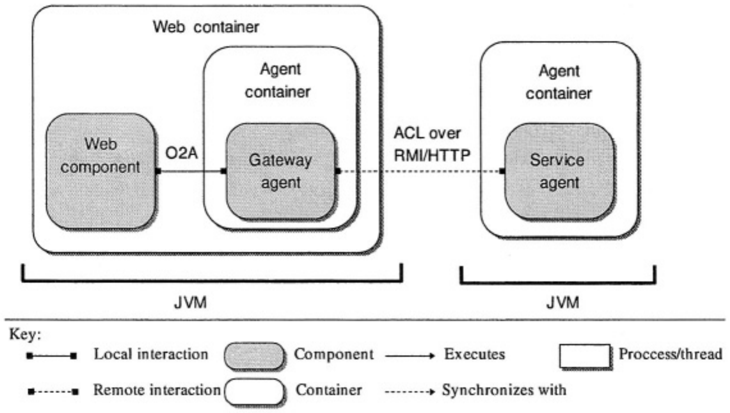


Fig. 1. Gateway Architecture. Component View

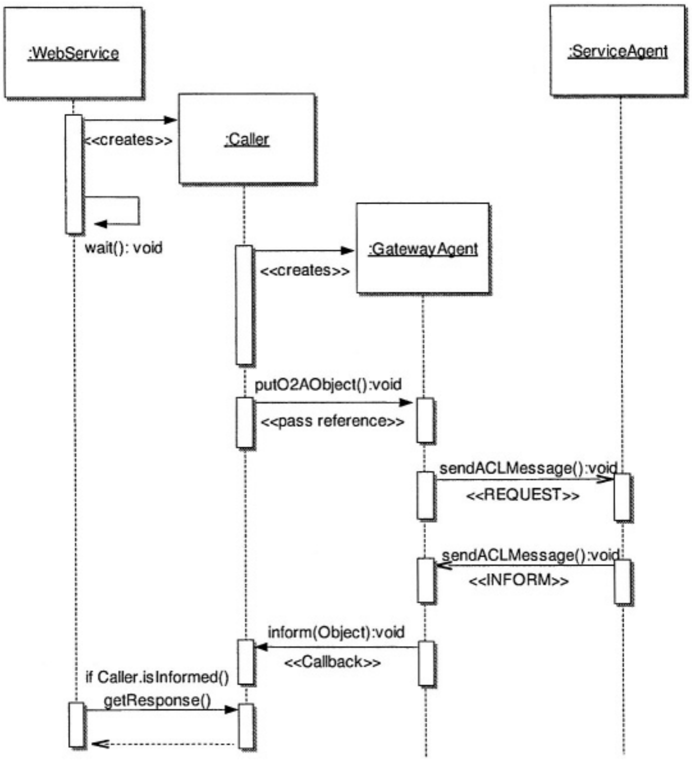


Fig. 2. Gateway Approach. Interactions Sequence Diagram

(even non Java-based ones) without access to its source code. As a consequence, the implementation efforts move into the web-tier, whose complexity required the project to rely on code generation tools.

2.2 Container Interactions, a Managed Applications Approach

Cowan and Griss [13], proposed BlueJADE, a connector that allows the management of a JADE platform by a Java application sever, and therefore its interaction with the enterprise applications deployed inside it. BlueJADE's main strengths occur at the container level. Although the containers are running separately, the connector eliminates the need for remote calls which enables the use of the object-agent channels. Figure 3 shows the coupling enhancement in the resulting architecture.

3 An Embedded Solution

3.1 Design Goals

In this work we are proposing an architecture for Web Service and agent integration that synthesizes and simplifies the preceding approaches in a way that achieves the goals of:

– Reducing the time to market of agent-based applications.
– Delivering good performance for large volume of users.
– Improving the code base maintainability.

This solution assumes that the agent and web component source code is available. Regarding the presented architectures, this solution mostly trades-off flexibility and manageability in favor of simplicity.

3.2 Component Model

In order to outline the present solution we started conceiving the agent-based application as a "black-box" which exposes several high-level web interfaces to their users. With respect to the gateway approach this idea arrives at a rather opposite approach that embeds the web container within the context of the Agent platform. As a result, by increasing internal cohesion and reducing coupling with respect of other applications, an agent system could be packed with their own specialized set of Web Services and operated via an embedded web application. The component view of the architecture its shown in figure 4.

Notice that some of the application server infrastructure present in BlueJADE architecture is replaced by lightweight custom programs. In this particular case, the *Launcher* is a main program, that initializes and starts the JADE platform besides an embedded instance of the Tomcat Web Server [14]. The *Registry* is nothing but a data structure that holds references to the running Service Agents implemented as a Singleton [15].

3.3 Interaction Model

At this point is easy to deduce how an interaction model could be without having to perform any remote method invocation. The model shown in figure 5 considers interactions similar to the ones in the gateway architecture until the

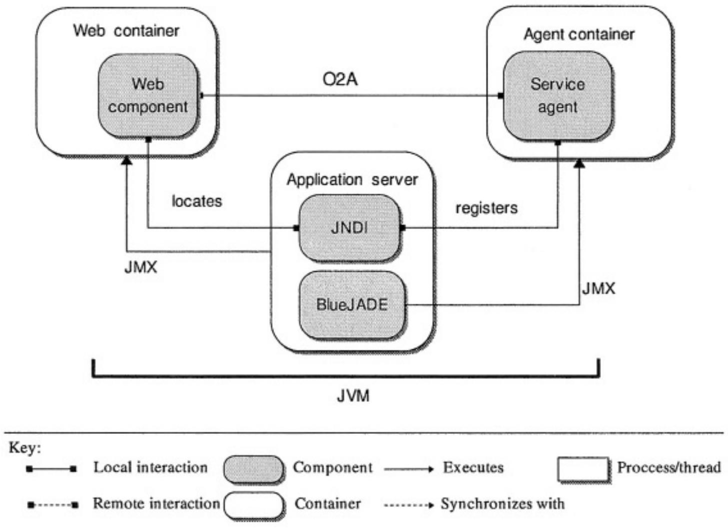


Fig. 3. Managed Applications Approach. Component View

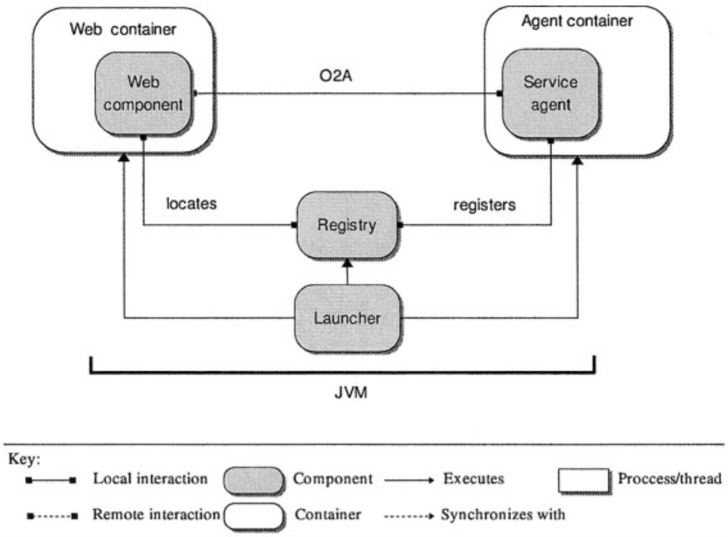


Fig. 4. Embedded Services Architecture. Component View

HTTP request is delivered to the Gateway Agent . In this embedded architecture the complex issues of interactions are handled between two core framework components:

– A *call monitor* object that serves as a shared memory space between the agent and the web component, and handles synchronization among them.

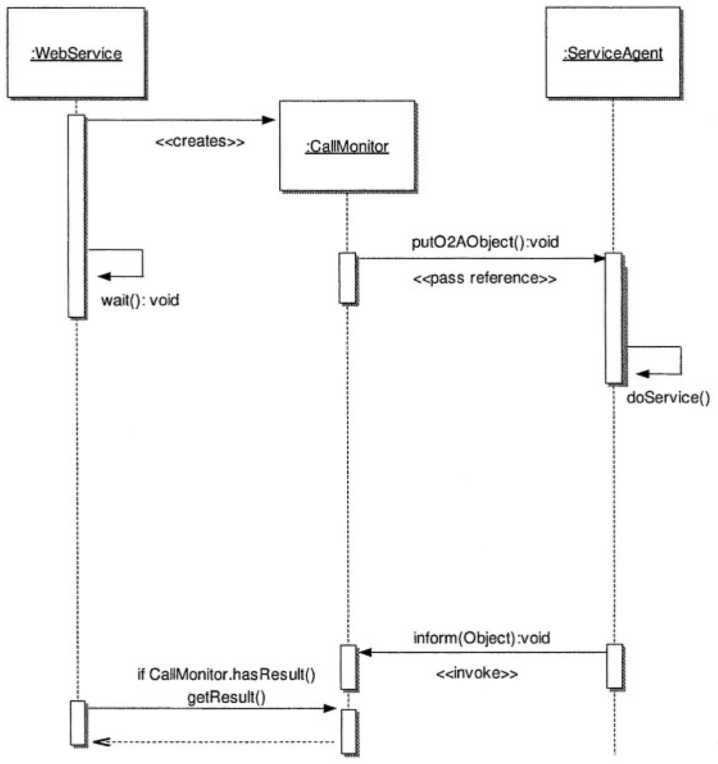


Fig. 5. Embedded Services Architecture. Interactions Sequence Diagram

– A *service behaviour* object used by agents for exposing internal tasks as Web Services.

4 Evaluation

By the means of simple metrics we could show some of the desirable properties of the proposed architecture. As seen in table 1, a functional implementation for embedding Web Services into agent based applications is several times smaller than the implementation of a gateway framework from the WSAI project.

Altough the number of classes is not an absolute metric of software complexity, from the interaction models we can deduce that the interactions present in the embedded model (figure 5) are nothing but a subset of the interactions required in a gateway approach (figure 2) , as a consecuence the internal complexity of the embedded Web Services architecture (EWSA) is not significatively different from the gateway architecture. In the embedded architecture only one abstract class is provided as extension point for service developers, which consequently simplifies the agent and Web Service implementation as shown in table 2.

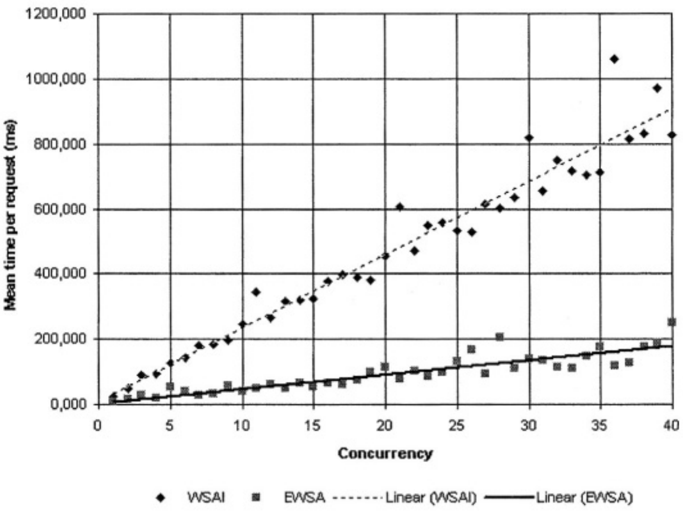


Fig. 6. Mean Service Time for Concurrent Requests

Table 1. Integration framework size comparison

Package	*Total classes*	*Abstract*	*Concrete*
Embedded	6	1	5
Gateway	52	6	46

This comparison is provided as a reference of the resulting programming models, considering that even if a code generation tool could actually simplify the development process, the additional interactions remains with a significative performance penalization.

A benchmark between the WSAI [11] sample currency exchange service and the same service implemented in the embedded architecture shows a clear improvement both in performance and in scalability. The performance gain in the embedded architecture can be interpreted as a result of the elimination of the nested network calls and their respective serialization and transmission overhead. Test results are shown in figure 6.

4.1 Requirements

In enterprise environments the problem of Web-enabling MultiAgent Systems may be stated in terms of the required interactions of software agents, web components, and their respective containers.

Although the fact that we consider specific products to implement the solution, in its more general form the proposed integration architecture could be implemented under several basic assumptions about the technologies used to produce Web-enabled MultiAgent Systems, noticeably:

Table 2. Example implementation size comparison

Package	*Total classes*	*Abstract*	*Concrete*
Embedded	2	0	2
Gateway	4	1	3

– Components and containers are implemented in a general purpose, interpreted[2],object-oriented language with thread support such as Java, Python or C#.
– For the language selected there exists an *agent container* that conforms to the FIPA Abstract Architecture [16] and Agent Management specifications. For the Java language we tested the container provided by the JADE[2] agent platform.
– *Service agents,* defined as those that process requests formulated by a human user or application, are implemented in a *thread-per-agent* model, in which agents periodically execute a set of tasks or behaviours as described by Rimassa[17].
– Agents have two communication queues or "virtual channels" that enables them to process FIPA ACL[18] messages or plain (Java) objects.
– There is a *web container,* that provides the execution environment for the web components. We integrated the Tomcat Web Server[14] that conforms to the Java Servlet API formally described in the JSR-154[8] specification.
– There is a well specified user-defined *web component* programming model. We consider the use of "Servlets" as they are a standarized and mature technology for the Java platform.

The embedded implementation we provide is useful in any Java based agent development environment, whenever the agent and web component source code is available. Adaptation for other languages and environments could be possible, though we don't delve into this question here.

5 Conclusions

In this paper we have outlined a simple and efficient architecture for embedding Web Services into agent-based applications. We presented experimental evidence to support our claims about simplicity and efficiency.

The embedded architecture synthesizes key features of other architectural proposals into an effective solution for developing web-enabled agent-based applications. It has been successfully used in "real-world" scenarios like the JITIK[3] Project, that is already supporting several collaborative environments.

We are starting a work on methodologies for using the proposed architecture in such a way that hybrid agent-web systems could be produced in a systematic way.

[2] That relies on platform-specific interpreters or virtual machines.
[3] Just-in-time information and knowledge[19], `http://lizt.mty.itesm.mx/jitik/`

References

1. Labrou, Y., Finin, T., Peng, Y.: Agent communication languages: the current landscape. IEEE Intelligent Systems **14** (March-April 1999) 45 –52
2. Bellifemine, F., Poggi, A., Rimassa, G.: JADE - A FIPA-compliant agent framework, `http://jade.cselt.it/papers/PAAM.pdf`. In: Proceedings of PAAM'99, London. (1999)
3. World Wide Web Consortium (W3C): Extensible Markup Language (XML) 1.0 (Second Edition), `http://www.w3c.org/TR/2000/REC-xml-20001006` (2000)
4. World Wide Web Consortium (W3C): Web Services Glossary, Working Draft, `http://www.w3c.org/TR/ws-gloss/` (Aug 2003)
5. World Wide Web Consortium (W3C): Web Services Description Language (WSDL) 1.1, `http://www.w3c.org/TR/wsdl` (2001)
6. World Wide Web Consortium (W3C): Simple Object Access Protocol, `http://www.w3.org/TR/SOAP` (2003)
7. Fielding, R.T., Taylor, R.N.: Architectural styles and the design of network-based software architectures. PhD thesis (2000)
8. Sun Microsystems, Inc.: JSR-000154 Java(TM) Servlet 2.4 Specification (Final release), `http://jcp.org/aboutJava/communityprocess/final/jsr154/index.html` (2003)
9. Sun Microsystems, Inc.: J2EE(TM) v1.3 Glossary, `http://java.sun.com/j2ee/1.3/docs/glossary.html` (2003)
10. Nwana, H.S., Ndumu, D.T.: A Perspective on Software Agents Research, `http://citeseer.nj.nec.com/nwana99perspective.html`. The Knowledge Engineering Review **14** (1999) 1–18
11. Whitestein Technologies, A.G.: Web Services Agent Integration Project, `http://wsai.sourceforge.net` (2003)
12. Dale, J., Willmott, S., Burg, B.: (Agentcities: Building a global next-generation service environment)
13. Cowan, D., et al., M.G.: Making Software Agent Technology available to Enterprise Applications, `http://www.hpl.hp.com/techreports/2002/HPL-2002-211.pdf`. Technical Report HPL-2002-211, HP Labs Technical Reports (2002)
14. Jakarta Project - The Apache Software Foundation: The Tomcat Web Server v. 4.1, `http://jakarta.apache.org/tomcat` (2003)
15. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley (1995)
16. Foundation for Intelligent Physical Agents: FIPA Abstract Architecture Specification, `http://www.fipa.org/specs/fipa00001/` (2002)
17. Rimassa, G.: Runtime Support for Distributed Multi-Agent Systems, `http://jade.cselt.it/papers/Rimassa-PhD.pdf`. In: Ph. D. Thesis, University of Parma. (Jan 2003)
18. Foundation for Intelligent Physical Agents: FIPA Communicative Act Library Specification, `http://www.fipa.org/specs/fipa00037/` (2002)
19. Brena, R., Aguirre, J.: Just-in-Time Knowledge Flow for Distributed Organizations using agents technology. In: Knowledge Technologies 2001 Conference, Austin Texas. (Mar 2001)

Gaining Competitive Advantage Through Learning Agent Models*

Leonardo Garrido[1], Ramón Brena[1], and Katia Sycara[2]

[1] Tecnológico de Monterrey, Centro de Sistemas Inteligentes
{leonardo.garrido, rbrena}@itesm.mx
[2] Carnegie Mellon University, The Robotics Institute
katia@cs.cmu.edu

Abstract. We present our bayesian-modeler agent which uses a probabilistic approach for agent modeling. It learns models about the others using a bayesian mechanism and then it plays in a rational way using a decision-theoretic approach. We also describe our empirical study on evaluating the competitive advantage of our modeler agent. We explore a range of strategies from the least- to most-informed one in order to evaluate the lower- and upper-limits of a modeler agent's performance. For comparison purposes, we also developed and experimented with other different modeler agents using reinforcement learning techniques. Our experimental results showed how an agent that learns models about the others, using our probabilistic approach, reach almost the optimal performance of the oracle agent. Our experiments have also shown that a modeler agent using a reinforcement learning technique have a performance not as good as the bayesian modeler' performance. However, it could be competitive under different assumptions and restrictions.

1 Introduction

A key focus on modeling other agents is concerned with the prediction of the behavior of other agents (exploiting the internal models about the others' preferences, strategies, intentions, and so forth). Then, the modeler agent can use this prediction in order to behave in the best way according to his preferences.

Research on modeling other agents has been approached from different perspectives. Tambe et al [1] have proposed an approach for tracking recursive agent models based on a plan recognition task. Gmytrasiewicz [2] has presented the Recursive Modeling Method (RMM) which uses nested models of other agents, combining game-theoretic and decision-theoretic mechanisms. Suryadi and Gmytrasiewicz [3] have proposed the use of influence diagrams for learning models about other agents. Vidal and Durfee [4] have developed an algorithm in order to see which of the nested models are important to choose in an effective manner. These authors have also presented a framework for determining the complexities of learning nested models [5].

* This research has been supported in part by the ITESM Research Chair CAT-011.

There have been other papers sharing our view of learning agent models in an iterative way using a Bayesian approach, for instance: Gmytrasiewicz et al [6] have proposed a framework for Bayesian updating of agent models within the formalism of the RMM; Zeng and Sycara [7] have presented an experimental research where a buyer models the supplier under a Bayesian representation in Bazaar, a sequential decision making model of negotiation.

In [8], we reported some results on evaluating, in an experimental way, the advantage an agent can obtain by building models about the others' roles and strategies. We explored a range of strategies from least- to most-informed in order to evaluate the upper- and lower-limits of our modeler agent performance. Now, in this paper we present an empirical evaluation of our bayesian-modeler and an agent learning models about other agents using different reinforcement learning strategies.

In the following sections, we first give a quick review our experimental framework. We also review the basic non-modeling strategies for comparison purposes. Then, we present our bayesian-modeler and reinforcement-modeler agents. Then, we present our experimental scenarios and discuss the results we obtained. Finally, we present the conclusions of this paper.

2 Experimental Framework

We have implemented the *Meeting Scheduling Game (MSG)* as our experimental testbed which models some characteristics of the distributed meeting scheduling problem. Our main concerns creating this test bed were: to allow self-interested as well as cooperative behavior, show or hide players' private information, and define different players' roles and strategies.

In this game, a group of agents try to arrange a meeting in such a way that certain meeting slot is available for as many as possible players. So that each player tries to arrange a meeting at a convenient and free time slot with an acceptable *utility* for him.

Each player's *role* is defined by a preference profile which is coded as a calendar slot utility function, ranking each slot from the most preferable slot to the least preferable one. We have defined several agent roles. For example: The *Early-Rising* prefers the early hours of the day; The *Night-Owl* prefers the meetings to be scheduled as late as possible; The *Medium* prefers the meetings to be around noon; The *Extreme* prefers to have meetings early in the morning or late in the afternoon.

Player's *strategies* are rules that tell them what actions to choose at each decision point. Strategies can take into account only the own player's preference profile or they can even use models about the others. In the subsequent sections we will define several different strategies.

Since a combination of a role and a strategy defines a player's preferences and behavior, the conjunction role/strategy of a player is seen as his *personality* in the MSG.

Each player proposes a slot taken from his own calendar composed of a working day with eight hours. Each player's calendar is set at a specific *calendar density* which is the proportion of busy hours in the calendar. The goal of a player in the MSG is to accumulate more points than his competitors in the game. A *game* consists of a predefined number of *rounds* and each player tries to accumulate points after each round.

There is a *referee* who ensures that all the players obey the rules of the game. He is also responsible for accumulating points for each agent after each round in an individual point counter for each player through the whole game.

After each round, each player's calendar is randomly reset, scrambling free and busy slots, maintaining the same predefined calendar density. Then, another round is started and the process is repeated until the predefined number of rounds is accomplished. Note that this implies we are not really "schedulling" any meetings, as the winning slots does not stand from a round to the next.

In each round, every player simultaneously proposes a slot according basically to his own individual strategy and role. However, the players' proposals are not completely determined by their own personalities because some slots can be busy in their calendars. In the first round, each player randomly proposes an available slot. These initial random proposals are needed as a "bootstrap" for the collaborative strategy defined in the following section. The other strategies are not affected by this initial round, since this is the only round where nobody accumulate points.

After all the players make their proposals, several *teams* are formed. Each team is composed of all those players who proposed the same calendar slot. Then, each *team joint utility* is calculated, summing up all the team members' calendar utilities: $TJU(t) = \sum_{\forall m \in t} U_m(s_t)$. Here, t is a team, m is a member of the team, s_t is the slot proposed by members in t, U_m is the slot utility of member m. Finally, the round is won by the team which accumulates the greatest team joint utility.

Once the winning team is selected, each agent earns points according to the following predefined *scoring procedure:* all the players outside the winning team accumulate zero points for that round and each agent a in the winning team t accumulates his own slot utility plus the team joint utility: $G_a(s) = TJU(t) + U_a(s_t)$.

Although this game is based on the general distributed meeting scheduling problem, it resembles only some of its characteristics.

3 Basic Strategies

We set up a framework for characterizing all the possible strategies in the MSG, ranging from a least-informed to the most-informed one. This allows us to place every given strategy in a framework where it can be better compared to others, and in particular to place modelling strategies in context. The lower and upper limits of our framework are given by the following strategies:

Indifferent Strategy: An agent using this strategy chooses his next proposal among his action set using an uniform (equiprobable) distribution.

Oracle Strategy: An agent using this strategy can see in advance the others' next move because he knows the other agents' calendars, roles and strategies. For each free slot s in his calendar, he calculates his possible gain $G_o(s)$, if he proposed that slot. Then, he finds the agent m who would earn the maximum gain $G_m(s)$ among the rest of the players, if he proposed that slot. Then, he calculates the utility of each slot s as his gain with respect to the profit of agent m: $\mathbf{U}(s) = G_o(s) - G_m(s)$. After checking all his free slots, he proposes the slot with the highest utility: $\arg\max_s \mathbf{U}(s)$.

An *indifferent agent* does not take into account any information about the other agents. He does not even take into consideration his own preferences. However, he must propose a free slot in his calendar, as must do all the other strategies as well. This strategy is considered as the lower limit for every "reasonable" strategy, since a strategy performing worse than the random is hardly worth considering.

An *oracle agent* knows the roles and strategies of the other agents (i.e. he has the correct models about the others). Furthermore, he even knows the others' calendars. So that an oracle agent is able to see in advance the others' moves and then he just chooses to propose the slot that maximizes his utility in each round of the game. Although an oracle agent has the best chances of winning each round, he can not always win! This is because of his random calendar availability, according to the fixed calendar density.

In order to have additional points of reference, we have also defined the following two heuristic strategies:

Self-Centered Strategy: This strategy tells the agent always to choose the free slot which just maximizes his own calendar slot utility.

Collaborative Strategy: Using this strategy, the agent chooses the free slot that was proposed by the biggest team (greatest number of members) at the previous round. In case of ties, the agent ranks them according to his own calendar slot utility.

A *self-centered agent* does not consider information about the other agents but he takes into account his role. A *collaborative agent* also takes into account the agent's own role. However, it also takes into consideration information about the previous round, trying to join in the biggest observed team.

4 Bayesian Strategy

Let us first introduce our term *model* about another agent. We just see it as a vector which records a probability distribution of the actual character of the modeled agent. In the context of the MSG, each agent has two basic models about each other agent a. The first one is the *role model:* $\mathbf{r}_a \overset{\text{def}}{=} (r_1, \ldots, r_n)$. Here, each r_i is the probability that agent a has the particular role i and n is the amount of different predefined roles. The notation $r_a(i)$ refers to the probability r_i of role i. The second model used in the MSG is the *strategy model:* $\boldsymbol{s_a} \overset{\text{def}}{=} (s_1, \ldots, s_m)$. Here, each s_i is the probability that agent a has strategy i and m is the amount of different predefined strategies. The notation $s_a(i)$ refers to the probability s_i of strategy i.

Since we are assuming independence between roles and strategies in the MSG (section 2), it is easy to construct a new combined model for each other agent: the *personality model.* This model is just a two-dimensional matrix, $\boldsymbol{rs_a}$, where each element $rs_a(i, j)$ is just calculated as follows: $rs_a(i, j) = r_a(i)s_a(j)$. Now, let us define an decision-theoretic strategy that take explicit advantage of knowing the others' models:

Semi-modeler Strategy: This strategy tells the agent to choose the slot which maximizes his expected utility based on predefined fixed models about the other agents.

It is assumed that a *semi-modeler agent* already have models about the others and his strategy just uses these probabilistic models to choose the action that maximizes his expected utility. The models are given to the semi-modeler agent at the beginning of the game and they never change during all the game. It is also important to note that the given models are not necessarily correct models about the others. In [9] we already presented a more detailed description of the semi-modeler agent.

In order to build a modeler agent, model construction is required. Let us define a modeler strategy that uses an Bayesian updating mechanism in order to build the others' models in an incremental and iterative way:

Bayesian-Modeler Strategy: An agent using this strategy incrementally builds models about the others using a Bayesian belief updating approach and chooses the action which maximizes his expected utility:

A *bayesian-modeler agent* does not have any information about the others. However, as stated in section 2, the set of predefined roles and strategies are public knowledge. At the beginning, the modeler agent can behave as a semi-modeler agent with equiprobable models about the others. That is, with no other knowledge about the others, it is reasonable to start with equiprobable probability distributions of the possible traits about the others. Then, the modeler agent can start to update those models based on the others' behavior.

This agent builds models about the other agents in an incremental and iterative way, updating those models after each round during the whole game. All the probabilities of each model are incrementally updated, trying to reach the actual character of the agent being modeled. The detailed bayesian-modeler strategy is as follows:

1. At the first round, start with equiprobable models about the others, run the semi-modeler strategy, and propose the resulting slot.

2. At the next round, for each other agent a:
   (a) Observe what was the a's proposal, s_a, in the previous round and update a's personality model, $\boldsymbol{rs_a}$, using a Bayesian updating mechanism to obtain the corresponding posterior probabilities of the a's personality, $per_a(i,j)$, given that a proposed slot s_a, $pro_a(s_a)$, in the previous round: $rs_a(i,j) = P(per_a(i,j)|pro_a(s_a))$.
   (b) Decompose the updated a's personality model in order to build two new separated role and strategy models. That is, update each element in $\boldsymbol{r_a}$ and $\boldsymbol{s_a}$: $r_a(i) = \sum_{\forall j} rs_a(i,j)$ and $s_a(j) = \sum_{\forall i} rs_a(i,j)$.
3. Using the new updated models about the others, run the semi-modeler strategy and propose the slot s_m with the maximum expected utility.
4. If it was the last round, the game is over. Otherwise go to step 2.

The model-updating mechanism is based on the well known *Bayes' rule.* In this case,we multi-valued random variables: the personality models. In fact, a personality model $\boldsymbol{rs}$ represents a probability distribution of personalities. So that the probability that an agent a has the personality resulting from combining role i and strategy j, $P(per_a(i,j))$, is precisely the value $rs_a(i,j)$ in matrix $\boldsymbol{rs_a}$ and the equation used to update each personality model can be rewritten as follows: $rs_a(i,j) = \frac{P(pro_a(s_a)|per_a(i,j))P(per_a(i,j))}{\sum_{\forall x}\sum_{\forall y} P(pro_a(s_a)|per_a(x,y))P(per_a(x,y))}$. The prior probabilities $P(per_a(i,j))$ are taken from the last recorded value $rs_a(i,j)$ in matrix $\boldsymbol{rs_a}$. The conditional probabilities $P(pro_a(s_a)|per_a(i,j))$ can be calculated from the known calendar density and the known agent behavior due to the personality $per_a(i,j)$. Thus, the bayesian-modeler is able to get all the posterior probabilities from the calculated conditional probabilities and the known prior probabilities. Then, this $\boldsymbol{rs}$ matrix is updated with these new probabilities in order to be used as prior probabilities in the following round.

5 Reinforcement Strategy

As in the case of our bayesian-modeler agent, our reinforcement-based strategy keeps and incrementally updates models about the others:

Reinforcement-Modeler Strategy: This strategy learns models about the others' personalities (i.e. roles and strategies). This strategy is very similar, in essence, to our bayesian-modeler strategy: it learns the others' models and exploits them with a greedy decision-theoretic approach.

The idea is to keep vectors of values instead of probabilistic vectors. Keeping this difference in mind, we have here three models for each other agent a as in the bayesian-modeler strategy case. The reinforcement-modeler agent constructs the state signal. That is, each state is not given just by the immediate sensations but it is a highly processed version of the sensations. Let us see then how we visualize the reinforcement learning in this case:

Discrete Time Steps: These correspond to the rounds of each game. That is, we have 10 time steps $t = 1, 2, ..., 10$.

States: At each time step t (at the beginning of each round), the agent receives a representation of the environment $s_t \in S$ where S is the set of possible states. In this case each state is the set of calendars with random free and busy slots of the other n agents plus the personality models of those n agents which are constructed by the reinforcement-modeler agent.

Actions: Based on the state, the learner will choose an action $a_t \in A(s_t)$ from a set of possible actions $A(s_t)$ available in state s_t. In this case each possible action a_t is just the proposal of each free slot in the calendar of the agent.

Rewards: As a consequence of its action, in this case the reinforcement-modeler agent will have $n \cdot m$ rewards, $r_{i,j,t+1}$, for each other agent i (of the n agents) with m different possible personalities. These rewards are not directly given by the referee agent but they are computed by the reinforcement-modeler using the information of each state.

Intuitively, if an agent chooses the first slot in the morning, we have a general human tendency of giving more weight (or reward) to an early-rising role than a night-owl one. Inspired by this fact, the rewards used by this strategy are not just the earned points by the agents at each round. Here, the rewards are a function of the observed proposed slot in the previous round. Thus, the strategy first observe the others' choices and then calculate the rewards for each possible personality of the other agents.

After updating all the personalities values, the reinforcement-modeler agent decomposes these models in two new separated role and strategy models for each other agent. This decomposition is the reverse process of the personality model composition explained above. In a similar way to the bayesian-modeler agent, here the reinforcement-modeler agent compute each element in $\boldsymbol{r_a}$ and $\boldsymbol{s_a}$ asfollows: $r_a(i) = \sum_{\forall j} rs_a(i,j)$ and $s_a(j) = \sum_{\forall i} rs_a(i,j)$. After this, the agent just re-uses the semi-modeler strategy in order to exploit the updated models, choosing the slot with the maximum expected utility.

6 Experimental Results

Here, what we call an *experiment* is a series of games with the same characteristics and groups of different and related experiments are called *experimental scenarios.* At the beginning of each game, all the agents are initialized with random roles taken from a set of two opposite roles (the early-rising and night-owl roles presented in section 2) and eight-slots calendars with the calendar density fixed at 50%. All the games are composed of ten rounds (the fourth and fifth experimental scenarios are the exceptions). Also in all these experiments, we run three agents (the exception is the second experimental scenario). Furthermore, when we run a bayesian-modeler or a reinforcement-modeler agent, he is always learning the models about the others and playing the game at the same time. We have set up series of games in order to measure how agent performance is affected by different strategies. Once a game is completed, we call it a "success", if the strategy under consideration wins. Otherwise it is considered a "failure". Our experiments are composed of 500 independent games and we have calculated that the results obtained in these experiments has a 95% confidence of getting an error not greater than about 0.05. In all tables presented here, we show the performance of each strategy as the percentage of success.

The goal of the first scenario (see table 1) is to compare the performance of the non-modeling strategies discussed in section 3. Thus, we run here an indifferent agent first against self-centered agents, then against collaborative ones, and finally against both. As expected, the performance of the indifferent strategy is always the worst, giving us a lower-limit performance to compare other reasonable strategies. We intuitively thought that the performance of the collaborative agents should be better because they can team each other. However, as we can see, in the first two experiments, the self-centered strategy appears to be better than the collaborative one against the indifferent agent. In the last, experiment, we can see that the self-centered strategy clearly outperforms the collaborative one, while the indifferent's performance is very low. As it is shown elsewhere [10], when incrementing the number of agents, the collaborative's performance increases, outperforming the self-centered.

Table 1. Comparing the basic strategies

Experimental Scenario 1			
Experiments	**Strategies**		
	Indifferent	*Self-Centered*	*Collaborative*
Experiment 1.1	7.59%	92.41%	—
Experiment 1.2	18.15%	—	81.85%
Experiment 1.3	3.86%	80.59%	15.45%

The goal of the second experimental scenario (see table 2) is to compare the performance of the bayesian-modeler agent discussed in section 4 and the semi-modeler strategy presented in 4 using fixed opposite, equiprobable and correct models about the others. Here we run four experiments with a self-centered agent, a collaborative one, and we vary the strategy of the third agent in each experiment. In the first experiment we run an oracle agent who has the correct models about the others. In the second one, we run a semi-modeler agent who uses fixed equiprobable models. In the third experiment, we again run a semi-modeler agent but now with fixed opposite models about the others. In the last one, we finally run a bayesian-modeler who is learning the models and playing at the same time during the ten rounds of each game. In the first experiment of this scenario, we get the empirical upper-limit performance given by the oracle strategy. On the other hand, running a semi-modeler with the incorrect fixed opposite models, we expect to have a lower-limit performance. We can see, in the third experiment, that this limit is indeed so low, being even the self-centered strategy the winner. The second experiment, shows a new more refined lower-limit performance, given by a semi-modeler with fixed equiprobable models. So that, our expectations for a good modeler performance is to get a performance somewhere between the upper-limit given by the oracle and the lower-limit given by the semi-modeler with fixed equiprobable models. As we can see, our expectations were confirmed in the last experiment.

Table 2. Comparing the bayesian and semi-modeler strategies

Experimental Scenario 2				
Exp	Strategies		Modeling	
	Self-Centered	*Collaborative*	*Models*	*Perform.*
2.1	20.58%	10.88%	Correct	68.54%
2.2	33.13%	11.31%	Equiprobable	55.56%
2.3	51.96%	20.19%	Opposite	27.85%
2.4	30.31%	7.67%	Bayesian	62.02%

The goal of the third scenario (see figure 1) is to evaluate the performance of both the bayesian-modeler and the reinforcement-modeler agent, varying the number of rounds needed to learn the models about the others in each experiment. Looking at the results in this scenario, it is clear how the performance improves as the number of rounds increases. Here, we can directly compare the different performances we have obtained with the indifferent, oracle, semi-modeler, bayesian-modeler and reinforcement-modeler strategies when playing against the self-centered and collaborative ones. As we can observe in games with only one round, the bayesian-modeler strategy performance starts with a performance between the limits of the semi-modeler strategies using fixed opposite and equiprobable models. This performance increases when we increase the number of rounds in the games, trying to reach upper-limit given by the oracle strategy. On the other hand, we can observe that the performance of the reinforcement-modeler is actually slightly worse than, but very close to, the bayesian-modeler one. Both performances increases as the number of rounds increases, showing a performance very close to the oracle one just few rounds after the first ten rounds. In fact, in other experiments, not shown here, we run games as long as 100 rounds which show that the reinforcement-modeler performance is slightly increasing trying to reach the asymptotic upper-limit performance but always lower than the performance of the bayesian-modeler.

7 Conclusions

In this paper, we presented our bayesian-modeler agent which models other agents in a probabilistic way, combining Bayes and decision theory, and we also presented an empirical evaluation of its performance. We have used a collection of reference points: The indifferent and oracle strategies provide the extremes of the spectrum, ranging from least- to most-informed strategies. We have also obtained other more refined reference points given by the semi-modeler strategy with fixed opposite and equiprobable models. Our experimental results have shown

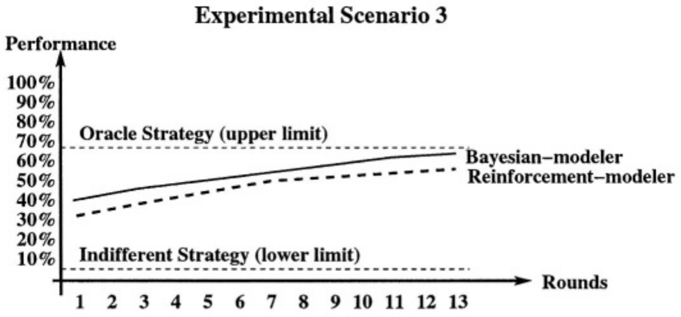


Fig. 1. Comparing the bayesian and reinforcement strategies

how both, the bayesian-modeler and reinforcement-modeler strategies performances, are indeed better than the empirical lower-limit we have obtained and, in fact, we have also observed how these performances increase as the number of rounds increases. Our experiments have also shown that after thirteen rounds the bayesian-modeler performance is really close to the oracle one. On the other hand, we could also observe how the reinforcement-modeler agent performance is increasing but it ends about 10% below the performance of the bayesian-modeler after about 13 rounds. However, this performance is considerable close to the optimal performance marked by the oracle agent.

References

1. Tambe, M., Rosenbloom, P.: Architectures for agents that track other agents in multi-agent worlds. In: Intelligent Agents II. Lecture Notes in Artificial Intelligence (LNAI 1037). Springer Verlag (1996)
2. Gmytrasiewicz, P.: On reasoning about other agents. In: Intelligent Agents II. Lecture Notes in Artificial Intelligence (LNAI 1037). Springer Verlag (1996)
3. Suryadi, D., Gmytrasiewicz, P.: Learning models of other agents using influence diagrams. In: IJCAI-99 Workshop on Agents Learning About, From, and With other Agents, John Wiley and Sons (1999)
4. Vidal, J., Durfee, E.: Using recursive agent models effectively. In: Intelligent Agents II. Lecture Notes in Artificial Intelligence (LNAI 1037). Springer Verlag (1996)
5. Vidal, J., Durfee, E.: Agents learning about agents: A framework and analysis. In: AAAI-97 Workshop on Multiagent Learning. (1997)
6. Gmytrasiewicz, P., Noh, S., Kellogg, T.: Bayesian update of recursive agents models. Journal of User Modeling and User-Adapted Interaction **8** (1998) 49–69
7. Zeng, D., Sycara, K.: Bayesian learning in negotiation. Internation Journal in Human-Computer Systems **48** (1998) 125–141
8. Garrido, L., Sycara, K., Brena, R.: Quantifying the utility of building agents models: An experimental study. In: Learning Agents Workshop at the Fourth International Conference on Autonomous Agents (Agents 2000). (2000)

9. Garrido, L., Brena, R.and Sycara, K.: On measuring the usefulness of modeling in a competitive and cooperative environment. In: Collaborative Learning Agents Workshop at the AAAI 2002 Spring Symposium. (2002)
10. Garrido, L., Brena, R., Sycara, K.: Towards modeling other agents: A simulation-based study. In Sichman, J., Conte, R., Gilbert, N., eds.: Multi-Agent Systems and Agent-Based Simulation. Volume 1534 of Lecture Notes in Artificial Intelligence. Springer-Verlag (1998) 210–225

Towards an Efficient Rule-Based Coordination of Web Services*

Eloy J. Mata[1], Pedro Álvarez[2], José A. Bañares[2], and Julio Rubio[1]

[1] Department of Mathematics and Computer Science, University of La Rioja, Edificio Vives, Luis de Ulloa s/n, E-26004 Logroño (La Rioja, Spain)
{eloy.mata, julio.rubio}@dmc.unirioja.es

[2] Department of Computer Science and Systems Engineering, University of Zaragoza, María de Luna 3, E-50015 Zaragoza (Spain)
{alvaper, banares}@unizar.es

Abstract. Web services coordination has become a central topic for further development of Internet-based distributed computing. One approach to this coordination task is supported by generative communication, and more specifically by some implementations of the Linda model as JavaSpaces. However, when applying these coordination strategies to real projects, some drawbacks appear. One of the main limitations is the lack of transactional queries. In this paper we deal with this problem extending the matching mechanism of the Linda model. Then, a variant of the well-known RETE algorithm can be devised in order to implement our extended Linda model efficiently. This also opens new research lines in which Artificial Intelligence techniques (as advanced blackboard architectures) could be applied to the field of web services coordination.

1 Introduction

From a conceptual point of view, most of the mechanisms used to coordinate web services may be interpreted as *message brokers:* the information items dispatched for coordination can be considered as messages, and then, the integration task itself can be seen as a logic to deliver messages, that defines implicitly a message broker. Routing logics can be classified according to the sender's identity, message type, or message content. They can be typically defined in a rule-like language. In this way, message brokers provide a loosely coupled architecture for application integration [1].

The shared dataspace of the Linda model [2] is particularly well-suited for this interpretation, and several extensions of it have been proposed in order to introduce the concept of coordination into the development of middleware for web-based environments [3]. The use of a shared dataspace for entity communication was first introduced in Artificial Intelligence through the notion of *blackboard* [4]. Blackboards are information spaces where messages can be published and retrieved.

* Partially supported by the Spanish Ministry of Science and Technology through projects TIC2002-01626, ACPI02-06 and TIC2003-09365-C02-01.

The main aim of this paper is to explore more deeply this relationship between Coordination and Artificial Intelligence by means of an extension of the Linda model providing a matching process with multiple templates.

The paper is structured as follows. Section 2 shows the differences between ruled-based languages and other coordination languages based on pattern matching. Section 3 describes a formal framework for modelling an extension of the tuple-based coordination language Linda. Section 4 presents a description of a Web Coordination Service (WCS). We propose an adaptation of the RETE algorithm to implement an extension of the matching operation with multiple templates. The classical example of the search and reservation of a set of travel tickets is used to show the role of a shared dataspace with a rule-based language. Finally, conclusions and future work are presented.

2 Pattern Matching-Based Coordination

Coordination deals with the management of dependencies between activities and the specification of the conditions under which a certain service may o may not be invoked. Different coordination models have been proposed in the literature, for example Petri nets [5], rules [6], Linda [2], or Statecharts [7].

Similarities between these coordination models stem from the use of templates to recognize particular states and represent constraints, and therefore the use of a pattern matching algorithm to interpret the models. The similarity between Petri nets and rules has already been pointed out in the literature (see [8, 9]), particularly by comparing the process of pattern matching [10, 11]. On the other hand, Linda is based on a blackboard used as a shared dataspace. Data are recovered by means of pattern matching operations.

In spite of the similarities between these models based on pattern matching, it is important to note the differences between rule-based languages and other coordination languages. Pure coordination languages are used to model process creation and coordination, procedures which are concepts orthogonal to computational languages. Therefore the difference is clear: a rule-based language represents a computational model that expresses the control flow by means of a pattern-directed matching algorithm, whereas coordination languages may use a matching algorithm to express the order and constraints in which “components” are involved in computation.

Another important difference between rule-based languages and pure coordination models stems from their different origins. Rule-based languages, as a knowledge representation language, put the emphasis on their expressiveness and have evolved towards the integration of different paradigms, while pure coordination languages propose simple operations independent of any computational language. In particular, this difference is reflected on the complexity of their corresponding matching operations. The main emphasis in rule-based languages has been to express sophisticated patterns to recognize different states, whereas in Linda, the main emphasis has been to keep simple operators, and therefore a simple matching process.

The expressiveness of rule-based languages has inspired a different paradigm, the event-condition-action (ECA) paradigm, when they are used to monitor the occurrence of *events* of interest. Once an event is received, an *action* is performed if a *condition,* i.e., a Boolean predicate over event parameters, is satisfied. The main drawback of rule-based languages is that big set of rules are difficult to understand and keep. It is important to note that when rule-based languages are used, there is not a clear control flow. However, this characteristic makes them suitable for defining exception-handling logic.

As an alternative approach, this paper increases the complexity of the Linda matching operation in order to enhance the expressiveness of Linda primitives, and therefore opens the possibility to express more sophisticated coordination restrictions. This extension is inspired by rule-based languages, but a proper extension on the Linda model keeping simple primitives is proposed. From this point of view, the extension is similar to High Level Petri nets, which provide more compact and manageable descriptions than ordinary Petri nets [12].

3 Extending the Linda Model to Web Services Coordination

Linda has been the first coordination model that actually adopted a dataspace with associative access for coordination purposes [2]. Linda introduces an abstraction for concurrent programming through a small set of coordination operations combined with a shared dataspace containing tuples. The *out* operation emits a tuple into the tuple space. The *in* and *rd* operation retrieves a tuple. A matching rule governs tuple selection from the tuple space in an associative way. In [13] a formal framework for tuple-based coordination models is developed; the tuple-based coordination medium is represented as a software component interacting with the coordinated entities by receiving *input events* and sending *output events.* The main ingredients of this model are: a set of tuples t ranging over *T;* a set of templates *templ* ranging over *Templ;* a matching predicate *mtc(templ,* t) between templates and tuples; and a choice predicate $\mu(templ, \bar{t}, \hat{t})$ extracting a tuple $\hat{t}$ from the multiset of tuples $\bar{t}$ that matches a template *templ,* or returning an error element $\perp_T$ if no matching is available. This is formally defined as $\frac{mtc(templ,t)}{\mu(templ,t|\bar{t},t)}$ and $\frac{\nexists\ t\in\bar{t}\ \ mtc(templ,t)}{\mu(templ,\bar{t},\perp_T)}$.

The status of a tuple space at a given time is characterized as a labelled transition system by the couple $\langle\bar{t},\overline{w}\rangle$ where $\bar{t}$ is a multiset of tuples and $\overline{w}$ is a multiset of *pending queries.* A pending query is an input event *ie* ranging over $IE \subseteq W$, where W is a set of operations such as the reading operation *rd,* the reading and removing operation *in,* or the writing operation *out,* which are possibly waiting to be served by a tuple space. Output events *oe* range over *OE,* with the syntax $\underline{o}v$ representing a message v for entity o.

The semantics of a tuple space is defined by the couple $\langle S, E\rangle$. $S \subseteq W \times \overline{T}$ is a satisfaction predicate for queries, so that $\langle w,\bar{t}\rangle \in S$ means w is satisfiable under the current space's content $\bar{t}$. $E \in W \mapsto 2^{(\overline{T}\times\overline{W})\times OE\times(\overline{T}\times\overline{W})}$ is a evaluation function, so that $\langle\bar{t},\overline{w},\widehat{oe},\bar{t}',\overline{w}'\rangle \in E(w)$ means that the evaluation of the pending query w causes the tuple space in state $\langle \bar{t}, \overline{w} \rangle$ to move to $\langle \bar{t}', \overline{w}' \rangle$ and produce output event $\widehat{oe}$ (or nothing).

Predicate S and evaluation function E are defined as the least relation satisfying a set of rules for every pending query corresponding to each primitive operation allowed. For instance, in the case of the read operation *rd(templ)°*, the rules $\frac{mtc(templ,t)}{S(rd(templ)^o,t|\bar{t})}$ and $\frac{mtc(templ,t)}{\langle t|\bar{t},\overline{w},\underline{o}t,t|\bar{t},\overline{w},\rangle \in E(rd(templ)^o)}$ must be satisfied.

In order to introduce Linda into a web-based environment, we work with a version of Linda where tuples and templates admit descriptions as sequences of *attribute/value* pairs, which is similar to the "top level" structure of any XML document. Now, in this XML-like context, Linda admits a more complex matching mechanism providing a promising way to coordinate, communicate and collaborate on the net [14].

Furthermore, we extend the Linda model provided in [13] by adding operations with multiple templates where two or more tuples are involved in, rd_2, in_2, $rd_1.in_1$, $rd_2.in_1$, and so on. The aim is to enrich Linda with some transactional capabilities (see an example in Subsection 4.3). For instance, the double tuple reading operation $rd_2(templ_1, templ_2)^o$, which is sent by the coordinated entity o that wants to read two tuples from the tuple space so that both tuples match both templates and share some common attributes. In this case, we need an extended matching predicate mtc_2 between two templates and two tuples. The predicate is defined as:

$$mtc_2(templ_1, t_1, templ_2, t_2) = mtc(templ_1, t_1) \wedge mtc(temp_2, t_2)$$

under the constraint:

$$select(templ_1, t_1, a) = select(templ_2, t_2, a) \quad \forall a \in sharing(templ_1, templ_2)$$

where $sharing : Templ \times Templ \longrightarrow List(A)$ is a list of common attributes and $select : Templ \times T \times A \longrightarrow V$ gives the value v of the attribute a of template $templ$ provided by a tuple $t \in T$.

Also, we can consider a predicate $\mu_2(templ_1, templ_2, \bar{t}, \widehat{t_1|t_2})$ by choosing two tuples $\widehat{t_1|t_2}$ which match two templates, $templ_1$ and $templ_2$, from a tuple multiset $\bar{t}$ defined as (with $\perp_T^2 = \{\perp_T, \perp_T\}$):

$$\frac{mtc_2(templ_1, t_1, templ_2, t_2)}{\mu_2(templ_1, templ_2, t_1|t_2|\bar{t}, t_1|t_2)}$$

$$\frac{\nexists\ t_1 \in \bar{t} \quad or \quad \nexists\ t_2 \in \bar{t} \quad mtc_2(templ_1, t_1, templ_2, t_2)}{\mu_2(templ_1, templ_2, \bar{t}, \perp_T^2)}$$

Semantics of rd_2 is described by the following rules where S is the satisfaction predicate and E is the evaluation function:

$$\frac{mtc_2(templ_1, t_1, templ_2, t_2)}{S(rd_2(templ_1, templ_2)^o, t_1|t_2|\bar{t})}$$

$$\frac{mtc_2(templ_1, t_1, templ_2, t_2)}{\langle t_1|t_2|\bar{t}, \overline{w}, \underline{o}t_1|\underline{o}t_2, t_1|t_2|\bar{t}, \overline{w} \rangle \in E(rd_2(templ_1, templ_2)^o)}$$

note that we should introduce also operations to read $n + m$ tuples and only remove the last m's: $rd_n.in_m(templ_1, \ldots, templ_n, templ_{n+1}, \ldots, templ_{n+m})^o$.

4 Design and Implementation

As it has been mentioned, it is necessary to make Linda primitives accessible through Internet protocols (HTTP, SMTP, SOAP), encode data in XML format, and integrate reactive behavior in order to introduce Linda into a web-based environment. In this Section, the design and implementation of a Web Coordination Service (WCS) based on the Linda model is presented. It plays the role of a broker of messages into the development of a middleware for web-based environments.

The implementation is based on a implementation of Linda called *JavaSpaces Technology* [15]. In JavaSpaces, a collection of processes may cooperate through the flow of Java objects (representing tuples) in a network-accessible shared space. Besides, the *Jini Distributed Event* model is incorporated into JavaSpaces for firing events when objects that match templates are written into a space. This allows to react to the arrival of input events when they are placed in the space.

First, Subsection 4.1 shows implementation details to provide Linda primitives through web protocols and extend the simple JavaSpaces matching operation to XML data. A more detailed explanation of this implementation may be found in [16, 17]. Then, in Subsection 4.2, a structure inspired by the RETE algorithm is proposed to support the extended matching based on multiple templates similar to rule-based languages presented in Section 3.

4.1 Extending JavaSpaces to Support XML Tuples, Web Protocols and Reactive Behavior

The WCS is composed of three software components (see Figure 2):

The **Web Coordination** component operates as a simple web-accessible interface of the Java Coordination component, providing the same collection of operations than JavaSpaces through web protocols (HTTP, SMTP, SOAP).

The **Java Coordination** component is the core of the service. It provides the collection of writing and reading operations proposed by Linda and allows processes to publish XML tuples and subscribe their interest in receiving XML tuples of a specific XML schema, encouraging a reactive style of cooperation between processes. This component is a repository of agents (here, an agent is a computational entity that acts on behalf of other entities in a semi-autonomous way and performs its tasks cooperating with other agents) that are capable of coordinating with other agents by exchanging XML tuples through XML-based spaces (more details in [17]).

Finally, the interactions between the web service and the applications occur in the **XML-based Space** component. XML data representing tuples are stored in JavaSpaces as Java objects. JavaSpaces provides a shared repository of Java objects and operations to `read` and `take` objects from and `write` them to it (they implement the *rd, in* and *out* Linda operations). But the matching rules of JavaSpaces must be extended to support this XML-based interoperability.

To solve this problem, the matching process is performed in two steps. In a first step, a matching based on XML-Schema (type of tuple), is performed by the original JavaSpaces matching primitive. In a second step, a particular matching method of the retrieved object that represents the same XML-Schema is invoked using the original template as a real parameter. This method checks if the field-values of the nodes are the same. Other approaches to enhance the matching process follow the same strategy. For example, in [18] to exchange encrypted data among distributed processes.

Note that this matching proposal in two steps does not guarantee the semantics of the Linda model. To solve this semantic problem, all the tuples with the same XML-Schema are stored in the same partition. A new Java object called *Channel* has been designed to manage partitions. It must guarantee a non-deterministic access to all the tuples stored in a given partition. Figure 2 shows an example of tuple-space partitioned by three different Channels. In [17] the efficiency of the proposed pattern is evaluated, and an additional partitioning of the interaction space to improve efficiency is proposed.

4.2 The Role of the RETE Algorithm

This Section presents the extension of the matching process to multiple templates formally introduced in Section 3 in a way similar to rule-based languages. This operation is the main source of inefficiencies in pattern-directed inference engines, and algorithms such as RETE [19] and TREAT [20] have been proposed to improve the efficiency of this process.

The underlying aim of these algorithms is to improve efficiency by matching only the changed data elements against the rules rather than repeatedly matching all the templates against all the data. With this purpose in mind, the information about previous matchings is recorded in a graph structure.

For example, the RETE network is composed of a global test tree common to all the rules, and a join tree specific to each rule. The test tree is composed of one-input nodes (called α-memories), each of them representing a test over some attribute. The information related to the binding consistency test is represented by the join tree associated to the rule. The join tree is composed of two-input nodes (called join nodes or β-memories). Join nodes collect data pointers for a consistent binding of template variables. The last join node collects all the data elements to which the associated rule can be applied. Figure 1 shows the RETE network resulting from the compilation of a sample rule in CLIPS syntax.

When a datum is added to the working memory, a new pointer is introduced into the root of the test tree and propagated if the test is successful.

In our case, there are two reasons to recover the RETE structure in order to support our extended matching: 1) to provide an efficient algorithm to support the proposed extended matching; and 2) to use the RETE structure as a base to support the semantics of the extended matching operation

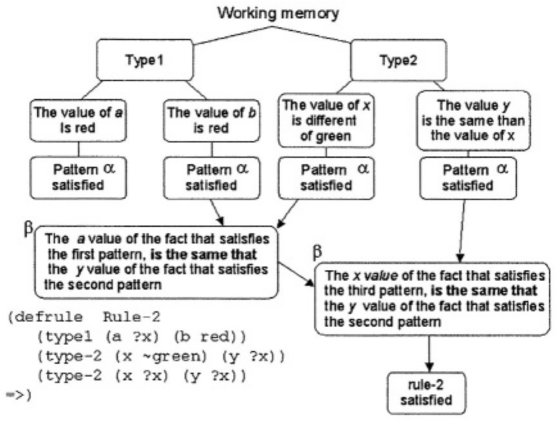

Working memory
Type1
Type2
The value of a is red
The value of b is red
The value of x is different of green
The value y is the same than the value of x
Pattern α satisfied
Pattern α satisfied
Pattern α satisfied
Pattern α satisfied
β
The a value of the fact that satisfies the first pattern, is the same that the y value of the fact that satisfies the second pattern
β
The x value of the fact that satisfies the third pattern, is the same that the y value of the fact that satisfies the second pattern
rule-2 satisfied
(defrule Rule-2
(type1 (a ?x) (b red))
(type-2 (x ~green) (y ?x))
(type-2 (x ?x) (y ?x))
=>)

Fig. 1. A sample of a RETE Network

with several templates. To this end, the last join nodes associated with each reading operation with several templates collect all the data that satisfy the templates, and also the blocked processes waiting for data (tuples) that match these templates.

When a reading operation with several templates is performed, an α-memory is created for each template (see Figure 2). Each α-memory is subscribed to the corresponding Channel (XML-Schema) and filters the data that match the template. Intermediary β-memories collect data pointers for a consistent binding of a template; and the last β-memory collects all the possible combinations of tuples in the XML Space that satisfy the operation. An alternative approach is to save only the last β-memory as in the TREAT algorithm.

If there is a set of tuples that matches the reading template, it will be found and propagated until the corresponding β-memory; otherwise, the reading process will be blocked. Reading processes are blocked until a set of tuples reaches the last β-memory. Then, the reading of all the tuples must be performed as an atomic action, i.e. through a rd_n operation, because a concurrent operation may remove some of the tuples that match the templates.

The JavaSpaces Technology uses the Jini Transaction model, which provides a generic transaction service for distributed computing. To perform all the reading operations as atomic actions, all of them are performed within the same transaction context. If all the reading operations are successful, the waiting process will be unblocked. Otherwise, if a reading operations fails, the transaction will be automatically aborted and the waiting processes will be blocked until a new set of tuples reaches the last β-memory.

As it has been pointed out in Section 3, a semantic aspect must be considered when various templates are used in an atomic action and only a subset of tuples matching them must be removed (i.e. through a $rd_n in_m$ operation). Then, it is necessary to mark the set of templates that imply the removal of the tuples that match them. This templates will be translate into the corresponding `take` operations, and the rest into `read` operations over JavaSpaces. The use of the

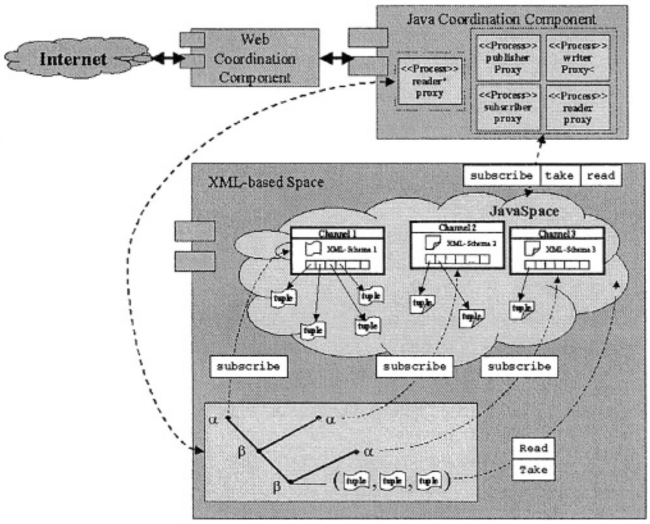

Internet
Web Coordination Component
Java Coordination Component
<<Process>> reader* proxy
<<Process>> publisher Proxy
<<Process>> writer Proxy<
<<Process>> subscriber proxy
<<Process>> reader proxy
XML-based Space
subscribe
take
read
JavaSpace
Channel 1
Channel 2
Channel 3
tuple
subscribe
subscribe
subscribe
α
β
Read
Take

Fig. 2. Use of a RETE Network to support the extended matching

support of JavaSpaces for transactions prevents any inconsistency. Once a tuple is read under a transaction, it cannot be involved in a new transaction until the first transaction is completed.

Finally, it is important to note the difference between our approach and the classical use of the RETE algorithm. In classical rule-based languages, rules are defined and compiled to generate the RETE network from the beginning and the structure remains immutable. In our proposal, each reading operation implies building a temporal structure that will be removed. The structure only remains if a subscription operation is performed. On the other hand, the main reason for the original RETE network is its efficiency in the pattern matching process. In our proposal, the network and its Channels provide a way to filter tuples and perform a test of binding consistency.

4.3 Example of the Use of Extended Matching

A classical example is the search and reservation of a set of tickets to travel from an origin to a destination. Let's assume that airlines introduce in a Web Coordination Service free seats to be booked, whereas railway companies publish timetable information. If a client is looking for train or fly tickets with intermediary stopovers tickets should be removed and train combinations should be read. However, conversations (protocols) between clients and companies in order to reserve and pay reservations is out of the scope of this paper. The following multiple template reading operation represents the constraints imposed:

```
read{*[(type fly)(from, London) (to,?FirstPort)
      (depart ?depart&After(?depart 8:30 0:00))
      (arrive ?arriveFirstPort) (ticket_number ?tickectfirstfly1)]
*[(type fly)(from, ?FirstPort) (to,Madrid)
 (depart ?depart&After(?depart ?arriveFirstPort 2:00))
 (arrive ?arriveSecondPort) (ticket_number ?tickectfirstfly2)]
 [(type train) (from, Madrid) (to,Zaragoza)
 (depart ?depart&After(?depart ?arriveSecondPort 1:00))
 (ticket_number ?tickectfirstfly2)]
}
```

The asterisk before the template denotes that the tuple should be removed. In our implementation, a tuple that must be removed will be translated into a JavaSpaces `take` operation. Otherwise, it will be translated into a `read` operation. And all the JavaSpaces operations involved in a reading operation with several templates will be performed within the same transactional context.

5 Conclusions and Further Work

In this paper an extension of the Linda model aimed at improving its usability in real applications of web services coordination has been presented. In a rather unexpected way, it has been also found a parallelism between generative communication and rule-based systems. A well-known technique taken from Artificial Intelligence, namely the RETE algorithm, can be used to obtain an efficient implementation of our theoretical proposal.

In order to introduce our ideas, a formal approach has been adopted. The relevance of this formal model, which is only partially sketched here, is twofold: on the one hand, it facilitates a proper extension of the Linda model, and not only a more o less fuzzy variant of it (this goal is relevant due to the existence of technological tools based on Linda as JavaSpaces); on the other hand, the formal model is simple enough to devise cleaner algorithm (in fact, the role of the RETE algorithm was discovered in this way).

Apart from the attribute/value, or more generally XML enhancement (which was previously introduced elsewhere [14]), our main contribution is the transactional capabilities supplied to the Linda model. Even if limited, these capabilities can be implemented on JavaSpaces and be used in non-trivial real applications, as shown in Subsections 4.2 and 4.3, respectively.

Further work should be done in order to fully implement our extended Linda model by including our RETE-like algorithm. In addition, more studies are needed in order to link the general field of web services coordination with advanced blackboards architectures and other related Artificial Intelligence areas.

References

1. Alonso, G., Casati, F., Kuno, H., Machiraju, V.: Web Services. Concepts, Architectures and Applications. Springer Verlag (2004)
2. Carriero, N., Gelernter, D.: Linda in context. Communications of the ACM 32 (1989) 444–458

3. Papadopoulos, G.A., Arbab, F.: Coordination models and languages. In: 761. Centrum voor Wiskunde en Informatica (CWI), ISSN 1386-369X (1998) 55
4. Englemore, R.S., Morgan, A.J., eds.: Blackboard systems. Addison-Wesley, Reading (MA) (1988)
5. Murata, T.: Petri nets: Properties, analysis and applications. Proceedings of the IEEE **16** (1990) 39–50
6. Kappel, G., Lang, P., Rausch-Schott, S., Retschitzegger, W.: Workflow management based on objects, rules, and roles. Data Engineering Bulletin **18** (1995) 11–18
7. Harel, D.: Statecharts: A visual formalism for complex systems. Science of Computer Programing **8** (1987) 231–274
8. Bruno, G., Elia, A.: Operational specification of process control systems: Execution of prot nets using OPS5. In: Proc. of IFIC'86, Dublin. (1986)
9. Duggan, J., Browne, J.: ESPNET: expert-system-based simulator of petri nets. IEEE Proceedings **135** (1988) 239–247
10. Valette, R., Bako, B.: Software implementation of Petri nets and compilation of rule-based systems. In: 11th International Conference on Application and Theory of Petri Nets, Paris (1990)
11. Muro-Medrano, P.R., Bañares, J.A., Villarroel, J.L.: Knowledge representation-oriented nets for discrete event system applications. IEEE Trans. on Systems, Man and Cybernetics - Part A **28** (1998) 183–198
12. Jensen, K., Rozenberg, G., eds.: High-level Petri Nets. Springer-Verlag, Berlin (1991)
13. Viroli, M., Ricci, A.: Tuple-based coordination models in event-based scenarios. In: IEEE 22nd International Conference on Distributed Computing Systems (ICDCS 2002 Workshops) - DEBS'02 International Workshop on Distributed Event-Based Sytem. (2002)
14. Álvarez, P., Bañares, J.A., Mata, E., Muro-Medrano, P., Rubio, J.: Generative Communication with Semantic Matching in Distributed Heterogeneous Environments. Number 2809 in Lecture Notes in Computer Science. In: Computer Aided Systems Theory – Eurocast 2003. Springer Verlag (2003) 231–242
15. Freeman, E., Hupfer, S., Arnold, K.: JavaSpaces. Principles, Patterns, and Practice. Addison Wesley (1999)
16. Álvarez, P., Bañares, J.A., Muro-Medrano, P., Nogueras, J., Zarazaga, F.: A Java Coordination Tool for Web-Service Architectures: The Location-Based Service Context. Number 2604 in Lecture Notes in Computer Science. In: Scientific Engineering for Distributed Java Applications. Springer Verlag (2003) 1–14
17. Álvarez, P., Bañares, J.A., Muro-Medrano, P.: An Architectural Pattern to Extend the Interaction Model between Web-Services: The Location-Based Service Context. Number 2910 in Lecture Notes in Computer Science. In: First International Conference on Service Oriented Computing –ICSOC 2003. Springer Verlag (2003) 271–286
18. Bettini, L., Nicola, R.D.: A Java Middleware for Guaranteeing Privacy of Distributed Tuple Spaces. Number 2604 in Lecture Notes in Computer Science. In: Scientific Engineering for Distributed Java Applications. Springer Verlag (2003) 175–184
19. Forgy, C.: A fast algorithm for many pattern / many object pattern match problem. Artificial Intelligence **19** (1982) 17–37
20. Miranker, D.P.: Treat: A better match algorithm for AI production systems. In: Proceedings of AAAI-87. (1987) 42–47

Applying Rough Sets Reduction Techniques to the Construction of a Fuzzy Rule Base for Case Based Reasoning

Florentino Fdez-Riverola[1], Fernando Díaz[1], and Juan M. Corchado[2]

[1] Dept. Informática, University of Vigo, Escuela Superior de Ingeniería Informática, Edificio Politécnico, Campus Universitario As Lagoas s/n, 32004, Ourense, Spain
{riverola, fdiaz}@uvigo.es
[2] Dept. de Informática y Automática, University of Salamanca, Plaza de la Merced s/n, 37008, Salamanca, Spain
corchado@usal.es

Abstract. Early work on Case Based Reasoning reported in the literature shows the importance of soft computing techniques applied to different stages of the classical 4-step CBR life cycle. This paper proposes a reduction technique based on Rough Sets theory that is able to minimize the case base by analyzing the contribution of each feature. Inspired by the application of the minimum description length principle, the method uses the granularity of the original data to compute the relevance of each attribute. The rough feature weighting and selection method is applied as a pre-processing step previous to the generation of a fuzzy rule base that can be employed in the revision phase of a CBR system. Experiments using real oceanographic data show that the proposed reduction method maintains the accuracy of the employed fuzzy rules, while reducing the computational effort needed in its generation and increasing the explanatory strength of the fuzzy rules.

1 Introduction and Motivation

Case-Based Reasoning (CBR) systems solve problems by reusing the solutions to similar problems stored as cases in a case base [1] (also known as memory). However, these systems are sensitive to the cases present in the case memory and often its accuracy depend on the significance of these stored cases. Therefore, in CBR systems it is always important to reduce noisy cases in order to achieve a good generalisation accuracy.

Case Based Reasoning is an application area where soft computing tools have had a significant impact during the past decade [2]. These soft computing techniques (fuzzy logic, artificial neural networks, genetic algorithms and rough sets, mainly) work in parallel for enhancing the problem-solving ability of each other [3].

In this paper we propose a reduction technique based on Rough Set theory applied to the revision stage of CBR systems. The proposed method was introduced into our Case-Based forecasting platform called FSfRT, specialized in making predictions for changing environments. Presently, the FSfRT platform is able to combine artificial neural networks and fuzzy logic in the framework of a CBR system.

The paper is structured as follows: section 2 introduces the Rough Set theory grounding; section 3 details the proposed Rough Set reduction technique; section 4 describes the Case-Based Reasoning platform used in this study; section 5 exposes the test bed of the experiments and the results obtained; finally, section 6 presents the conclusions and further work.

2 Rough Set Theory

Rough Set theory, proposed by Pawlak [4,5], is an attempt to provide a formal framework for the automated transformation of data into knowledge. It is based on the idea that any inexact concept (for example, a class label) can be approximated from below and from above using an indiscernibility relationship. Pawlak [6] points out that one of the most important and fundamental notions to the Rough Set philosophy is the need to discover redundancy and dependencies between features.

The main advantages of Rough Set theory are that it: (i) provides efficient algorithms for discovering hidden patterns in data; (ii) identifies relationships that would not be found while using statistical methods; (iii) allows the use of both qualitative and quantitative data; (iv) finds the minimal sets of data that can be used for classification tasks; (v) evaluates the significance of data and (vi) generates sets of decision rules from data.

2.1 Basic Concepts and Definition

Briefly, the relevant Rough Set terminology is stated below. An information system is a pair $S = \langle U, A \rangle$, where U is a non-empty and finite set, called the universe, and A is a non-empty, finite set of attributes (or features). An equivalence relation, referred to as indiscernibility relation, is associated with every subset of attributes $P \subseteq A$. This relation is defined as:

$$IND(P) = \{(x, y) \in U \times U : \text{for every } a \in P,\ a(x) = a(y)\}\,. \tag{1}$$

Given any subset of features P, any concept $X \subseteq U$ can be defined approximately by the employment of two sets, called lower and upper approximations. The lower approximation, denoted by $\underline{P}X$, is the set of objects in U which can be certainty classified as elements in the concept X using the set of attributes P, and is defined as follows:

$$\underline{P}X = \cup\{Y \in U / IND(P) : Y \subseteq X\}\,. \tag{2}$$

The upper approximation, denoted by $\overline{P}X$, is the set of elements in U that can be possibly classified as elements in X, formally:

$$\overline{P}X = \cup\{Y \in U / IND(P) : Y \cap X \neq \varnothing\}\,. \tag{3}$$

The degree of dependency of a set of features P on a set of features R is denoted by $\gamma_R(P)$, $0 \leq \gamma_R(P) \leq 1$, and is defined as:

$$\gamma_R(P) = \frac{card(POS_R(P))}{card(U)} . \tag{4}$$

where

$$POS_R(P) = \bigcup_{X \in U/IND(P)} \underline{R}X . \tag{5}$$

$POS_R(P)$ contains the objects of U which can be classified as belonging to one of the equivalence classes of $IND(P)$, using only features from the set R. If $\gamma_R(P) = 1$, then R functionally determines P.

Various extensions have been defined from the basic model proposed by Pawlak. Among these extensions stands out the Variable Precision Rough Set model (VPRS) [7] which is a generalisation that introduces a controlled degree of uncertainty within its formalism. This degree is established by an additional parameter ϕ.

2.2 Rough Sets as Reduction Technique

A major feature of the Rough Set theory is to find the minimal sets of data that can be used for classification tasks. In this sense, the notions of core and reduct of knowledge are fundamental for reducing knowledge preserving information. After stating the formal definitions of these concepts, it is outlined the reduction process proposed by the methodology.

P is an independent set of features if there does not exist a strict subset P' of P such that $IND(P) = IND(P')$. A set $R \subseteq P$ is a reduct of P if it is independent and $IND(R) = IND(P)$. Each reduct has the property that a feature can not be removed from it without changing the indiscernibility relation. Many reducts for a given set of features P may exists. The set of attributes belonging to the intersection of all reducts of P is called the core of P:

$$core(P) = \bigcap_{R \in Reduct(P)} R . \tag{6}$$

An attribute $a \in P$ is indispensable if $IND(P) \neq IND(P \setminus \{a\})$. The core of P is the union of all the indispensable features in P.

The reduction technique stated by the methodology is specially suitable for reducing decision tables. A decision table is an information system of the form $S = \langle U, A \cup \{d\} \rangle$, where $d \notin A$ is a distinguished attribute called the decision attribute or class attribute. The elements of the set A are referred to as condition attributes. A decision table is a classifier that has as its internal structure a table of labelled instances. Given a novel instance, the classification process is based on the search of all matching in-stances in the table. If no matching instances are found, unknown is returned; otherwise, the majority class of the matching instances is returned (there may be multiple matching instances with conflicting labels). The indispensable attributes, reducts, and core can be similarly defined relative to a decision attribute or output feature. The precise definitions of these concepts can be fount in Pawlak's book on Rough Sets [5].

At this point, it is very important to use the classification rules (given by a decision table) with the minimal effort, and therefore, the simplification of decision tables is of primary importance. The simplification process comprises two fundamental tasks. On the one hand, reduction of attributes consists of removing redundant or irrelevant attributes, without losing any essential classification information. The computation of the reducts for the condition attributes relative to the decision attribute is carried out to achieve this goal. On the other hand, reduction of attribute values is related to the elimination of the greatest number of condition attribute values, maintaining also the classificatory power.

3 Feature Subset Selection Using Rough Sets

The computation of the reducts and the core of the condition attributes from a decision table is a way of selecting relevant features. It is a global method in the sense that the resultant reduct represents the minimal set of features which are necessary to maintain the same classificatory power given by the original and complete set of attributes. A straighter manner for selecting relevant features is to assign a measure of relevance to each attribute and choose the attributes with higher values.

In the Rough Set framework, the natural way to measure the prediction success is the degree of dependency defined above. However, [8] have shown the weakness of this measure in order to assess an estimation of the predictive accuracy of a set of condition attributes Q with regard to a class attribute d. To overcome this deficiencies, [9] define the notion of rough entropy. Based on this notion and its adaptation to the VPRS model (in order to exploit more efficiently the knowledge that is provided for the observations in the boundary region or the uncertain area of the universe), we have defined a coefficient that allows to asses the significance of an attribute within a set of attributes [10]. The significance of an attribute $a \in Q$ is defined in a way that its value is greater when the removal of this attribute leads to a greater diminution of the complexity of the hypothesis $Q \setminus \{a\}$, and simultaneously, to a lesser loss of accuracy of the hypothesis. Implicitly, the underlying principle used to evaluate the relevance of an attribute in this way is the Minimum Description Length Principle (MDLP) [11].

The associated complexity of a given set of condition attributes Q can be evaluated through the entropy of the partition $U / IND(Q)$, which will be denoted by $H(Q)$. On the other hand, the conditional rough entropy $H_{\phi}(d \mid Q)$ can be used to evaluate the accuracy that is achieved when the condition attributes Q are used to predict the value of the condition attribute d. Therefore, the formal definition of the **ϕ-rough** entropy, denoted by $RH_{\phi}(d \mid Q)$, is given by the following expression:

$$RH_\phi(Q,d) = H(Q) + H_\phi(d \mid Q) = H(Q) +$$
$$\left\{ \{1 - \gamma_{Q,\phi}(d)\} \log_2 |U| + \sum_{x_i \subseteq \mathrm{POS}_{Q,\phi}(d)} \frac{|X_i|}{|U|} \log_2 \frac{|X_i|}{|U|} \right\}.$$
$$= \{1 - \gamma_{Q,\phi}(d)\} \log_2 |U| - \sum_{x_i \subseteq \mathrm{POS}_{Q,\phi}(d)} \frac{|X_i|}{|U|} \log_2 \frac{|X_i|}{|U|} \tag{7}$$

where X_i represents each one of the classes of the partition $U / IND(Q)$, the set $POS_{Q,\phi}(d)$ is the positive region of Q with regard to the decision attribute d, and $\gamma_{Q,\phi}(d)$ is the degree of dependence of attribute d on the set of attributes Q.

Then, the ϕ**-significance** of a condition attribute, $a \in Q$, with regard to the decision attribute d, denoted by $\sigma_{a,\phi}(Q, d)$, is defined as the variation that the ϕ**-rough** entropy suffers when the considered attribute is dismissed from Q. Namely, it is computed the term $\Delta_a RH_\phi(Q, d)$, given by the difference between $RH_\phi(Q, d)$ and $RH_\phi(Q \setminus \{a\}, d)$. Formally,

$$\sigma_{a,\phi}(Q,d) = \Delta_a RH_\phi(Q,d) = RH_\phi(Q,d) - RH_\phi(Q \setminus \{a\}, d)$$
$$= \{H(Q) - H(Q \setminus \{a\})\} - \{H_\phi(d \mid Q \setminus \{a\}) - H_\phi(d \mid Q)\} \quad . \tag{8}$$

Fig. 1 provides a concise description of the algorithm that selects a subset of relevant features using the significant ϕ**-rough** coefficient to evaluate the relevance of a feature. The proposed algorithm for selecting relevant features is described according to the view proposed by [12]. These authors state that a convenient paradigm for viewing feature selection methods is that of heuristic search, with each state in the search space specifying a subset of the possible features. Following Blum and Langley viewpoint the four basic issues that characterise this method are:

- The starting point in the space, which in turn influences the direction of search and the operators used to generate successor states. The proposed algorithm starts with all attributes and successively removes them (lines 1 and 15, respectively). This approach is known as backward elimination.
- The organisation of the search. Any realistic approach relies on a greedy method to traverse the space considering that an exhaustive search is impractical. At each point in the search, the proposed algorithm considers all local changes, namely, it evaluates the significance of each attribute of the current set of attributes (loop for).
- The strategy used to evaluate alternative subsets of attributes. In this paper, the variation of the normalised ϕ**-rough** entropy has been chosen for this purpose. Specifically, at each decision point the next selected state is that one which results from removing the attribute with the least significant ϕ**-rough** coefficient (line 10).
- A criterion for halting the search. In the algorithm, the criterion for halting is that the difference between the degree of dependency at initial state and the current one (both with respect to the decision) do not exceed a predefined threshold (line 14).

```
Function FEATURE_SUBSET_SELECTION (input: decisionTable, $\phi$; output: outputFeatures)
{
    begin.
    outputFeatures ← Features(decisionTable) /* the starting point is the complete set of input features */
    $\gamma_{initial}$ ← CoefDependence(decisionTable, outputFeatures, TargetFeature(decisionTable), $\phi$)
    step ← 0; haltCriterion ← FALSE
    while ¬haltCriterion do /* while it is not satisfied criterion of halt, remove features */
        haltCriterion ← TRUE; step ← step + 1
        $\sigma_{max}$ ← 1.0;
        for each feature f ∈ outputFeatures do /* select the most irrelevant feature */
            $\sigma_f$ ← CoefSignificance(decisionTable, f, outputFeatures, TargetFeature(decisionTable), $\phi$)
            $\gamma$ ← CoefDependence(decisionTable, outputFeatures \ {f}, TargetFeature(decisionTable), $\phi$)
            if ( $\sigma_f < \sigma_{max}$ ) then
                $\gamma_{current}$ ← $\gamma$
                feature_to_remove ← f
        $\gamma_{threshold}$ ← $\gamma_{initial}$*(1.0 – DecrementFactor(step));
        If ($\gamma_{current}$ >= $\gamma_{threshold}$ ) then
            outputFeatures ← outputFeatures \ {feature_to_remove} /* remove feature */
            haltCriterion ← FALSE
    end.
}
```

Fig. 1. Algorithm for feature subset selection

4 Description of the FSfRT Platform

The study described in this paper was carried out in the context of the FSfRT platform. FSfRT is a structured hybrid system that can employ several soft computing techniques in order to accomplish the 4-steps of the classical CBR life cycle [1]. This section covers two main points: (i) details the architecture of the FSfRT platform and (ii) introduces the use of Rough Sets inside the whole system.

4.1 FSfRT Platform Architecture

The FSfRT platform is an extension of a previous successful system [13] able to make predictions of red tides (discolourations caused by dense concentrations of microscopic sea plants, known as phytoplankton). The FSfRT platform allows us to combine several soft computing techniques in order to test their suitability working together to solve complex problems. The core and the interfaces of FSfRT have been coded in Java language and new capabilities are being developed. The general idea is to have different programmed techniques able to work separately and independently in co-operation with the rest. The main goal is to obtain a general structure that could change dynamically depending on the type of problem. Fig. 2 shows a schematic view of the system.

On the left of Fig. 2, it is shown the core of the platform that is composed by a KAM *(Knowledge Acquisition Module).* The KAM is able to store all the information needed by the different techniques employed in the construction of a final CBR system. In the retrieve and reuse stages, several soft computing techniques can be used [2,3], while in the revise stage, our platform employs a set of TSK fuzzy systems [14] in order to perform the validation of the initial solution proposed by the system.

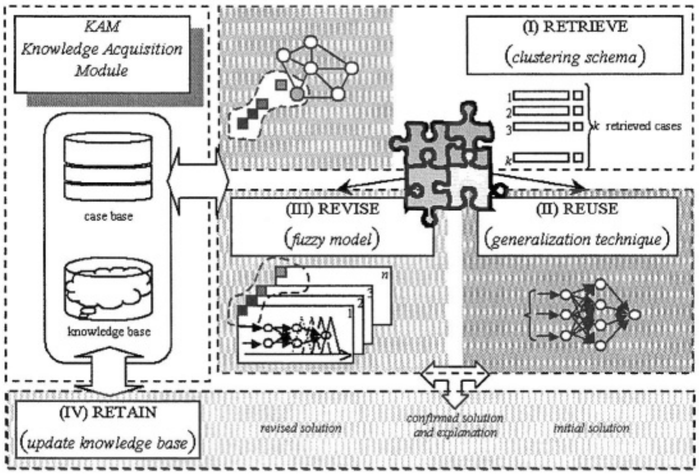


Fig. 2. FSfRT platform architecture

Our aim in this work, is to perform a feature subset selection step before the induction process carried out by the fuzzy revision method. The aim of this proposal is to reduce the original set of attributes and therefore decrease the computational effort for the generation of the different fuzzy models.

4.2 Rough Sets Inside the FSfRT Platform

Fig. 3 shows the meta-level process when incorporating the Rough Sets as a preprocessing step before the generation of the fuzzy revision subsystem.

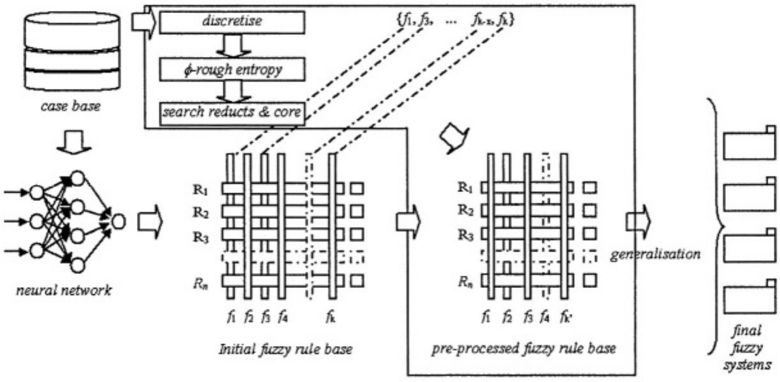


Fig. 3. Rough Set pre-processing step

For details related to the construction of the fuzzy systems starting from a Radial Basis Function neural network see [14]. The Rough Set process described here involves the first step in the generation of the initial fuzzy system and it is divided into three phases:

The first one discretises the cases stored in the case base. It is necessary in order to find the most relevant information using the Rough Set theory. The second one uses the significant **ϕ-rough** coefficient to select a subset of relevant features as described in section 3 (see Fig. 1). Finally, the last phase searches for reducts and core of knowledge from the features selected in the previous phase, as explained in section 2.

The motivation of including the second phase is that the computation of reducts is a blind technique, where several combinations of a sufficient number of irrelevant features can become a reduct. The pre-selection of features leads to reducts with a lesser complexity and a higher predictive accuracy.

5 Empirical Study

In order to evaluate the proposed method, we use a biological database composed by several physical variables (temperature, PH, oxygen, salinity, etc.) measured at distinct depths and belonging to different monitoring points of the north west coast of the Iberian Peninsula. These data values are complemented with data derived from satellite images stored separately. The satellite image data values are used to generate cloud and superficial temperature indexes. The whole memory of the system consists on approximately 6300 cases, each one represented as a feature vector that holds 56 attributes.

The FSfRT platform was configured to use the same techniques as in our previous work [14], where the fuzzy revision method was successfully tested: (i) a Growing Cell Structure (GCS) neural network as retrieval method, (ii) a Radial Basis Function (RBF) neural network for the reuse step and (iii) the aforementioned set of TSK fuzzy systems working as the revision mechanism. Specific information about these techniques and its integration inside the CBR life cycle can be found in [15]. The main goal of the previous work was to develop a forecasting biological system capable of predicting the concentration of diatoms (a type of single-celled algae) in different water masses.

Although the experiments carried out in [14] showed the effectiveness and the straightforward improvement of the proposed fuzzy revision method over other approaches, some issues remained unsolved in order to deploy the application for real use. Concisely, the main drawbacks of the tested method were: *(i)* the time needed for generating each one of the TSK fuzzy models and *(ii)* the explanatory complexity of the fuzzy rules used for the final solution proposed by the system.

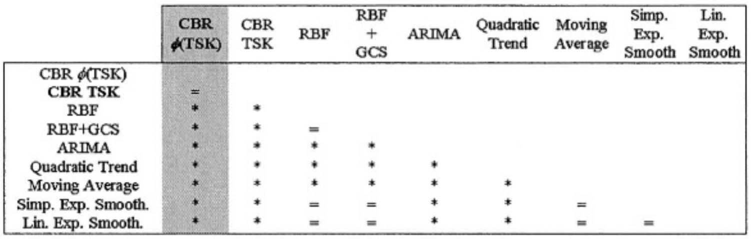
In order to solve these problems maintaining at the same time the accuracy level, we have proposed in this paper a feature subset selection algorithm based on Rough Set theory. As we can see in Table 1, several ϕ values have been tested in order to obtain the most accurate set of representative features defining each problem case. For the current domain of diatoms forecasting, the optimal number of features was 12 (ϕ = 0.01), corresponding to the physical magnitudes measured with a smaller level of depth and those generated from satellite images.

A crucial aspect in this experiment is the accuracy level of the Rough Set based revision subsystem and its comparison with the original one. Starting from the error

Table 1. Selected features depending on the value of the parameter ϕ

parameter ϕ	number of selected features
0.0	16
0.01	**12**
0.025	10
0.05	8
0.1	8

series generated by the different models, the Kruskall-Wallis test has been carried out. Since the P-value is less than 0.01, there is a statistically significant difference among the models at the 99.0% confidence level. Fig. 4 shows a multiple comparison procedure (Mann-Withney) used to determine which models are significantly different from the others. The experiments were made with a data set of 448 cases randomly taken from the case base. It can be seen that the CBR with TSK fuzzy revision subsystem (CBR TSK) presents statistically significant differences with the rest of the models, whilst it is as accurate as the simplified method presented here (CBR ϕ **(TSK)).**



	CBR ϕ(TSK)	CBR TSK	RBF	RBF + GCS	ARIMA	Quadratic Trend	Moving Average	Simp. Exp. Smooth	Lin. Exp. Smooth
CBR ϕ(TSK)									
CBR TSK	=								
RBF	*	*							
RBF+GCS	*	*	=						
ARIMA	*	*	*	*					
Quadratic Trend	*	*	*	*	*				
Moving Average	*	*	*	*	*	*			
Simp. Exp. Smooth.	*	*	=	=	*	*	=		
Lin. Exp. Smooth.	*	*	=	=	*	*	=	=	

Fig. 4. Mann-Withney test carried out between each pair of models

The time consumed in the execution of the pre-processing step plus the whole generation of the TSK fuzzy systems (2 hours aprox. in a Pentium IV processor) was the 80% less than the amount needed for generating the original fuzzy revision subsystem. This timesaving operation is motivated by the simplified fuzzy rule base employed by the greedy algorithm used to generate each one of the TSK fuzzy systems.

Another relevant circumstance derived from the adoption of the proposed schema was the increment in the explanatory strength of the justification generated by the final CBR system. Initially the feature vector describing a problem was composed of 56 attributes, the same as the fuzzy rule antecedents, now the system is able to produce an explanation based on only 12 main features with the same level of accuracy.

6 Conclusion and Further Work

This paper introduces a new reduction technique based on Rough Set theory that can be applied for improving a previous successful method that automates the revision stage of CBR systems.

Empirical studies show that this reduction technique allow us to obtain a more general knowledge of the model and gain a deeper insight into the logical structure of the system to be approximated. Employing the simplified fuzzy rule base as the starting point to generate the fuzzy revision subsystem proposed in [14], leads to a dramatic decrease of the time needed for this task while maintaining an equivalent generalisation accuracy.

These benefits are augmented with the simplicity of the new fuzzy rules used by the CBR system as explanation of the final adopted solution. In this way, it is interesting the definition of a formal measure in order to rate and compare the explanation strength of these fuzzy rules.

Due to the suitability showed by the Rough Set theory working together with other soft computing techniques, we are also interested in the development of new ways to put together this formalism with the existing techniques coded in the FSfRT platform.

References

1. Riesbeck, C.K., Schank, R.C.: Inside Case-Based Reasoning, Lawrence Erlbaum Associates, Hillsdale, NJ, US (1999)
2. Pal, S.K., Dilon, T.S., Yeung, D.S.: Soft Computing in Case Based Reasoning, Springer Verlag, London (2000)
3. Sankar, K.P., Simon, C.K.S: Foundations of Soft Case-Based Reasoning, Wiley-Interscience, Hoboken, New Jersey (2003)
4. Pawlak, Z.: Rough Sets. International Journal of Computer and Information Sciences, Vol. 11. (1982) 341–356
5. Pawlak, Z.: Rough Sets: Theoretical Aspects of Reasoning about Data. Kluwer Academic Publishers, Dordrecht (1991)
6. Pawlak, Z.: Rough sets: present state and the future. Foundations of Computing and Decision Sciences, Vol. 11 (3-4). (1993) 157–166
7. Ziarko, W.: Variable Precision Rough Set Model. Journal of Computer and System Sciences, Vol. 46. (1993) 39–59
8. Düntsch, I., Gediga, G.: Statistical evaluation of rough set dependency analysis. International Journal of Human-Computer Studies, Vol. 46. (1997) 589–604
9. Düntsch, I., Gediga, G.: Uncertainty measures of rough set prediction. Artificial Intelligence, Vol. 106. (1998) 77–107
10. Díaz, F., Corchado, J.M.: A method based on the Rough Set theory and the MDL principle to select relevant features. Proc. of the X CAEPIA - V TTIA, Vol. 1. (2003) 101–104
11. Rissanen, J.: Minimum description length principle. In Kotz, S. and Johnson, N. L. (eds.). Encyclopedia of Statistical Sciences. John Wiley and Sons, New York (1985) 523–527
12. Blum, A.L., Langley, P.: Selection of relevant features and examples in machine learning. Artificial Intelligence, Vol. 97. (1997) 245–271
13. Fdez-Riverola, F., Corchado, J.M.: FSfRT, Forecasting System for Red Tides. An Hybrid Autonomous AI Model. Applied Artificial Intelligence, Vol. 17 (10). (2003) 955–982
14. Fdez-Riverola, F., Corchado, J.M.: An automated CBR Revision method based on a set of β-TSK Fuzzy models. Proc. of the X CAEPIA - V TTIA, Vol. 1. (2003) 395–404
15. Fdez-Riverola, F.: Neuro-symbolic model for unsupervised forecasting of changing environments. Ph.D. diss., Dept. of Computer Science, Vigo University, Spain (2002)

Dynamic Case Base Maintenance for a Case-Based Reasoning System

Maria Salamó and Elisabet Golobardes

Enginyeria i Arquitectura La Salle, Universitat Ramon Llull,
Quatre Camins 2, 08022 Barcelona, Catalonia, Spain
{mariasal,elisabet}@salleurl.edu

Abstract. The success of a case-based reasoning system depends critically on the relevance of the case base. Much current CBR research focuses on how to compact and refine the contents of a case base at two stages, acquisition or learning, along the problem solving process. Although the two stages are closely related, there is few research on using strategies at both stages at the same time. This paper presents a model that allows to update itself dynamically taking information from the learning process. Different policies has been applied to test the model. Several experiments show its effectiveness in different domains from the UCI repository.

1 Introduction

Learning is a process in which an organized representation of experience is constructed [Scott, 1983]. However, this experience cause two problems in Case-Based Reasoning (CBR) systems, as reported in recent years. The first one is the *swamping problem* which relates to the expense of searching large case-bases for appropriate cases with which to solve the current problem. The second one is that the experience can be *harmful* and may degrade the system performance (understanding performance as problem solving efficiency).

Research on the area highlights to deal with negative knowledge using different strategies. Negative Knowledge is correct knowledge that can be a source of unsuccessful performance [Markovitch, S. and Scott, P.D., 1988]. Minton has demonstrated by *selective discarding knowledge* in a system [Minton, 1985] that the performance can be improved. Usually, the strategy of avoiding negative knowledge in the initial case base is not enough to achieve maximum performance for a CBR system. It is usually also necessary to integrate into the system a repeated maintenance during the problem solving process. There are several methods that fulfill these requirements, like competence-preserving deletion [Smyth and Keane, 1995], failure-driven deletion [Portinale et al., 1999], as well as for generating compact case memories [Smyth and Mckenna, 2001]. More close to our proposal are the one that examines the benefits of using fine-grained performance metrics to guide case addition or deletion [Leake and Wilson, 2000].

Previously to this paper, we have presented different approaches to case base maintenance [Salamó and Golobardes, 2003] in acquisition stage that allow us to reduce the case base in a controlled way and, at the same time, maintain the efficiency in the CBR system. Although our objectives had been achieved, our previous conclusions and the research on the area move us to go deeply into an extended treatment of the case base.

This paper introduces a dynamic case base maintenance (DCBM) model that updates the knowledge (case base in CBR) based on the learning problem solving process. The knowledge update is based on Reinforcement Learning. This approach can be considered as a "wrapper" model to case base maintenance. However, the authors propose it as a dynamic model because it depends completely on the problem solving process of the CBR system.

The paper is organized as follows. In section 2 we introduce the dynamic case base maintenance model and then different policies to apply it. Section 3 details the fundamentals of our experiments. Next section shows and analyzes the effectiveness of the model with the experimental results. Finally, we present the conclusions and further work.

2 Dynamic Case Base Maintenance

The foundation of our Dynamic Case Base Maintenance (DCBM) proposal is Reinforcement Learning. So, first we summarize its basis. Next, we describe how to use the Reinforcement Learning in our system, how the coverage of a CBR system can be modelled, and how different policies can exploit this model to perform a dynamic experience update able to control and optimize the case base while solving new cases.

2.1 Reinforcement Learning

Reinforcement Learning (RL) [Sutton and Barto, 1998] combines the fields of dynamic programming and supervised learning to yield powerful machine-learning systems. Reinforcement Learning appeals to many researchers because of its generality.

Reinforcement Learning [Harmon, 1996] is an approach to learning by trial and error in order to achieve a goal. A RL algorithm does not use a set of instances which show the desired input/output response, as do *supervised learning* techniques. Instead, a reward given by the environment is required. This reward evaluates the current state of the environment. The *Reinforcement Learning Problem* (RLP) consists of maximizing the sum of future rewards. The goal to be accomplished by RL is encoded in the received reward. To solve the problem, a RL algorithm acts over the environment in order to yield maximum rewards. Any algorithm able to solve the RLP is considered a RL algorithm.

Reinforcement Learning theory is usually based on *Finite Markov Decision Processes* (FMDP). The use of FMDP allows a mathematical formulation of the RLP, therefore, the suitability of RL algorithms can be mathematically proved.

Several elements appear in all RLPs. In each iteration the RL algorithm observes the *state* s_t of the environment and receives the *reward* r_t. The reward is a scalar value generated by the *reinforcement function* which evaluates the current state and/or the last executed action according to RLP. Following the rules of the RL algorithm, it generates an *action* a_t. The *environment* reacts to the action changing state s_t and generating a new state s_{t+1} . The *value function* contains the expected sum of future rewards. This function is used and modified by the RL algorithm to learn the policy function. A *policy function* indicates the action to be taken at each moment.

Initially, the approximation of the optimal value function is poor. Therefore, it is necessary to approximate the value function at every iteration. There are several methods that can be applied.

In order to find the optimal value functions, the Bellman equation is applied: $V^*(X_t) = r(X_t) + \gamma V^*(X_{t+1})$, where $V^*(X_t)$ is the optimal value function; X_t is the state vector at time t; X_{t+1} is the state factor vector at time $t+1$; $r(X_t)$ is the reinforcement function and γ is the discount factor in the range [0,1].

2.2 Dynamic Case Base Maintenance Model

There are several methodologies to solve the RLP formulated as a FMDP: dynamic programming, temporal difference algorithms and monte-carlo methods. We will use a Monte-Carlo method because is the only one that use experience of the environment to learn the value functions.

The question that arises now is how this idea can be applied to our model. Lets consider the model by analogy of the elements described in section 2.1. For our purpose a *state* s_t is a *case* of the environment that receives a *reward* r_t. The reward is a value generated by the *reinforcement function* which evaluates if the current state classifies or not classifies correctly. In our model the *reinforcement function* is the revise phase of the CBR cycle. Following the rules of the RL algorithm, which includes the case base maintenance policy, it generates an *action* a_t. The action for us is to delete or to maintain a case from the case base. The *environment* is the CBR cycle. The *environment* reacts to the action changing to state s_{t+1}, if the action is to delete the case. Thus, reducing the case base. The environment also generates a new reward after the problem solving process which has used the possibly reduced case base. The *value function* contains the expected sum of future rewards. This function is used and modified by the RL algorithm to learn the optimal case base. We test two different policy functions. Figure 1 shows the description of all the process. In our case, the RL algorithm receives a set of states and a reward for each one, and returns to the environment a set of actions.

Definition 1 (Coverage).
Let $T = \{t_1, t_2, ..., t_n\}$ be a set of training cases, $\forall\ t_i \in T$: $Coverage_k(t_i)$ will be the value of the metric used by the case base maintenance method at iteration k.

The *coverage* is the goodness value of a case when it is used to solve a target problem. It can be defined in several ways depending on the case base maintenance techniques used. For instance, it can be defined [Smyth and Keane, 1995] as the set of target problems that it can be used to solve. Here, we modify slightly the definition in order to adapt it to our model. The *coverage* is defined as the initial sum of future rewards using a Rough Sets measure. That is, $Coverage_k(t_i)$ is the value function at iteration k for state t_i.

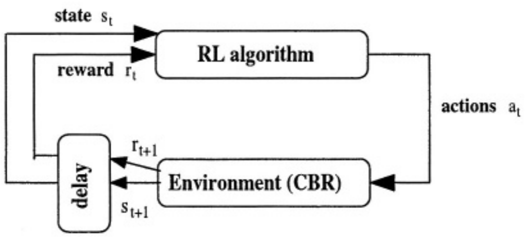


Fig. 1. Relation between RL algorithm and the environment

As detailed previously, the most important part of the RL algorithm is to update the value function. We use a Monte-Carlo (MC) which interacts with the environment following a particular policy function. In our model it is the optimizer of the case base. When the episode finishes, the MC algorithm updates the value of all visited states based on the received rewards. The visited states for a CBR cycle will be the *k*NN cases retrieved to solve the new problem. Equation 1 shows the general update rule to estimate the state-value function. Our MC algorithm is detailed in definition 2.

Definition 2 (CoverageUpdate).
Let $T = \{t_1, t_2, ..., t_n\}$ be a set of *K*NN cases, $\forall\ t_i \in T$:

$$Coverage_{k+1}(t_i) \leftarrow Coverage_k(t_i) + \alpha \cdot |R_t - Coverage_k(t_i)| \quad (1)$$

It can be observed that the current prediction of the state-value $Coverage_k(t_i)$ is modified according to the received sum of rewards R_t. The R_t value is 1.0 if the state t_i solve the target problem, otherwise it is 0. There is also a learning rate α which averages the values obtained in different episodes. The learning rate is usually set up to value 0.1 or 0.2 in RL systems. If the states are updated quite often it is set up to value 0.1, otherwise to 0.2. The selection of *K*NN neighbors in a CBR cycle may not often be repeated, so we have set up this learning rate to 0.2 in order to accelerate the differences of *Coverage* in few iterations.

Once described our value function update, we describe entirely the dynamic case base maintenance (DCBM) model in algorithm 1, which shows that the retrieval phase selects *K* Nearest-Neighbors, although it uses the best neighbor to solve the new problem. We consider the selection of *K*NN in order to accelerate the maintenance process of the case base. Another important point is the relation of the retain stage with the RL algorithm (step 9 and 10) in algorithm 1. The retain phase receives the set of actions to improve the case base.

The most notable aspect of the dynamic case base maintenance process is that the CBR system improves the case base using its problem solving process. Moreover, the case base improves or degrades the coverage of a case depending

Algorithm 1 Dynamic Case Base Maintenance (DCBM) Model

```
DCBM (CaseMemory $T$)
1.  Initialize $Coverage(t_i)$ using a CBM metric in acquisition stage, for all $t_i \in T$
2.  $T_{k+1} \leftarrow$ Reduce the initial case base $T_k$ using $Coverage$
3.  Repeat until problem solving process of the CBR cycle is not finished
4.  $T_k \leftarrow T_{k+1}$
5.    Retrieval phase $\leftarrow$ selects from $T_k$ the $K$NN used to solve the new problem
6.    Reuse phase $\leftarrow$ selects the best $1NN$ to solve the new problem
7.    Revise phase $\leftarrow$ computes the rewards $R_t$ of the $K$NN
8.    Retain phase $\leftarrow$ computes :
9.      CoverageUpdate $\leftarrow$ for each $t_i \in K$NN
10.     Apply case base maintenance policy function to decide the set of Actions $A$
11.     $T_{k+1} \leftarrow$ Update case base $T_k$ based on the Actions $A$
```

on their resolution accuracy. Thus, the case base can be categorized at different levels of coverage. The lower the coverage of a case, the most appropriate to disappear from the case base.

2.3 Dynamic Case Base Maintenance Policy Functions

The core of the RL process is the case base maintenance policy function. We describe two different policies to test the reliability of the proposed Dynamic Case Base Maintenance (DCBM) model.

RLOLevel. This policy is called Reinforcement Learning Oblivion policy by Level of Coverage (RLOLevel). This policy uses a similar philosophy that our acquisition [Salamó and Golobardes, 2003] case base maintenance method called ACCM. If we start from the premise that ACCM works well to reduce the case base while maintaining the prediction accuracy, it leads us to believe that the same process will be useful for dynamic maintenance. Thus, the complete process is detailed in algorithm 2.

Algorithm 2 RLOLevel

```
1. SelectCasesRLOLevel (CaseMemory $T$)
2. confidenceLevel = 1.0 and freeLevel = ConstantTuned (set at 0.01)
3. select all instances $t_i \in T$ as $SelectCase(t_i)$ if $t_i$ satisfies:
   $coverage(t) \geq$ confidenceLevel
4. while not $\exists$ at least a $t_i$ in $SelectCase$ for each class $c$ that $class(t_i) = c$
5.   confidenceLevel = confidenceLevel - freeLevel
6.   select all instances $t_i \in T$ as $SelectCase(t_i)$ if $t_i$ satisfies:
     $coverage(t_i) \geq$ confidenceLevel
7. end while
8. Action $A$ is to delete from CaseMemory $T$ the set of cases NOT selected as $SelectCase$
9. return Action $A$
```

The algorithm 2 tries to remove as much cases as possible. Therefore, the selection process is repeated until it accomplishes that every distribution class contains at least one case selected. Thus removing from case base those cases not selected. It is clear that this process will be very aggressive with the case base because it maintains the minimum description of the case base. It leads us to believe that this policy function may not work properly in a dynamic environment.

RLOCE. This policy is called Reinforcement Learning Oblivion by Coverage and Error (RLOCE). The coverage is the relevance of a case. This policy shows the simplest way to decide the actions.

```
Algorithm 3 RLOCE

1. SelectCasesRLOCE (CaseMemory $T$)
2. for each instance $t \in T$
3.   if $coverage(t) < initialCoverage(t)$ then $SelectCase(t)$ end if
4. Action $A$ is to delete those cases selected
5. return Action $A$
```

The policy is based on coverage lost. A case will be deleted if it classifies incorrectly new problems more often than correctly. Thus, the cases that produce misconception are deleted.

3 Description of the Experimental Analysis

This section is structured as follows: first of all, it is important to understand the fundamentals of our metric to initialize the coverage of a case. Then, we describe the testbed used and its characteristics. Finally, we analyze with different experiments the dynamic case base maintenance model.

3.1 Fundamentals

The rough sets theory defined by Pawlak, which is well detailed in [Pawlak, 1982], is one of the techniques for the identification and recognition of common patterns in data, especially in the case of uncertain and incomplete data. The mathematical foundations of this method are based on the set approximation of the classification space.

Each case is classified using the elementary set of features which can not be split up any further, although other elementary sets of features may exist. In the rough set model the classification knowledge (the model of the knowledge) is represented by an equivalence relation *IND* defined on a certain universe of cases U and relations (attributes) R. The pair of the universe cases U and the associated equivalence relation *IND* forms an approximation space. The approximation space gives an approximate description of any subset X of U. Two approximations are generated by the available data about the elements of the set X, called the lower and upper approximations. The *lower approximation* $\underline{R}X$ is the set of all elements of U which can *certainly* be classified as elements of X in knowledge R. The *upper approximation* $\overline{R}X$ is the set of elements of U which can *possibly* be classified as elements of X, employing knowledge R. In order to discover patterns of knowledge we should look for the minimal set of attributes that discerns cases and classes from each other, such a combination is called a *reduct.*

Measure of Relevance Based on Rough Sets. The reduced space, composed by the set of *reducts (P)* is used as a metric to extract the relevance of each case.

Definition 3 (Coverage Based on Rough Sets).
This metric uses the *quality of classification* coefficient, computed as:

$$\forall\, t_i \in T \textit{ it computes} : Coverage(t_i) = \frac{card\ (\ \underline{P}(t_i))\ \cup\ card\ (-\overline{P}(t_i))}{card\ (\ all\ instances)} \qquad (2)$$

Where $Coverage(t_i)$ will be the coverage of case t_i; T is the training set; *card* is the cardinality of a set; P is a set that contains the reducts; and finally $\underline{P}(t_i)$ and $\overline{P}(t_i)$ is the presence of t_i in the lower and upper approximation respectively.

The *Coverage* coefficient expresses the percentage of cases which can be correctly classified employing the knowledge t. This coefficient has a range of real values in the interval [0.0, 1.0]. Where 0.0 and 1.0 mean that the case is internal and outlier respectively.

We will use the *Coverage* as *initialCoverage* in our DCBM model. We also use the *Coverage* in our reduction technique (ACCM) in acquisition stage. Our experiments analyze the behaviour of DCBM model in front of ACCM. Our RLOLevel policy function is based on this algorithm. We apply ACCM in the training case base to select a range of cases that have to be deleted from the case base [Salamó and Golobardes, 2003]. ACCM maintains all the cases that are outliers, so cases with a *Coverage* = 1.0 value, and those cases that are completely internal, so cases with a *Coverage* near 0.0. Thus, reducing from the case base those cases that are not outlier and have a coverage near 1.0.

Using *coverage* values, we have two kind of cases relevant in the case base: the ones with coverage value of 1.0 (outliers) and the internal cases, having low coverage value. This coverage distribution is not much suitable for the RL policy functions which rely on high coverage values. Thus, we modify, previously to update phase and independently if we have applied ACCM or not, the *coverage* value with this formula: $Coverage(t) = 1 - Coverage(t)$, with the exception of outlier cases that have a $Coverage(t) = 1.0$. Therefore, we obtain *coverage* values that show relevance according to RL policy functions.

3.2 Testbed

The evaluation performance of the approaches presented in this paper is done using different datasets which are detailed in table 1. Datasets can be grouped in: *public* [Merz and Murphy, 1998] and *private* [Golobardes et al., 2002] that comes from our own repository. These datasets were chosen in order to provide a wide variety of sizes, combinations of feature types, and difficulty because some of them contain a great percentage of inconsistencies.

The percentage of correct classifications and the percentage of case base maintained has been **averaged** over **stratified ten-fold cross-validation** runs. To study the performance we use **paired** $\boldsymbol{t}$**-test** on these runs.

The study described in this paper was carried out in the context of our CBR system: BASTIAN (*case*-**BA***sed* **S***ys***T***em for class***I***fic***A***tio***N**). All techniques were run using the same set of parameters for all datasets: The case base is a list of cases. Each case contains the set of attributes, the class, the *Coverage* and

Table 1. Details of the datasets used in the experimental analysis

	Dataset	Ref.	Samples	Num. feat.	Sym. feat.	Classes	%Inconsistent
1	*Balance scale*	*BL*	625	4	3	2	2.0
2	*Breast cancer Wisconsin*	*BC*	699	9	-	2	0.30
3	*Credit-A*	*CA*	690	5	9	2	9.71
4	*Heart-H*	*HH*	294	6	7	5	20.4
5	*Heart-Statlog*	*HS*	270	13	-	2	0.0
6	*Hepatitis*	*HP*	155	6	13	2	0.0
7	*Horse-Colic*	*HC*	368	7	15	2	5.67
8	*Ionosphere*	*IO*	351	34	-	2	0.0
9	*Iris*	*IR*	150	4	-	3	0.0
10	*Labor*	*LB*	57	8	8	2	0.0
11	*Mammogram (private)*	*MA*	216	23	-	2	5.00
12	*Soybean*	*SY*	683	-	35	19	10.08
13	*TAO-Grid (private)*	*TG*	1888	2	-	2	0.0
14	*Vehicle*	*VE*	846	18	-	4	0.0
15	*Vote*	*VT*	435	-	16	2	4.13

the *initialCoverage*. Furthermore, the retrieval phase extracts the *K*-Nearest Neighbor to be updated in the RL process, not for the reuse phase which uses a **1-Nearest Neighbor.** We do not learn new cases during problem solving stage.

4 Analysing the DCBM Policy Functions

First of all, we test our DCBM policy functions using all the training set in front of 1NN algorithm and our reduction algorithm (ACCM) in acquisition stage (see columns 2 to 9 in table 2). We introduce in this experiment ACCM algorithm in order to compare the case base reduction (size) with our DCBM policies.

We observe that the best prediction accuracy is often obtained using oblivion by level of coverage (OL) and oblivion by coverage and error (OCE). Looking at ACCM algorithm, it has greater reduction than 1NN. In spite of the fact the reduction of DCBM policies is not as great as ACCM, because its selection to delete is founded on the *K*NN selected, they produce a good balance between reduction and improvement of prediction accuracy. That is, they are less aggressive reducing the case base than ACCM.

There is a clear conclusion: if we prefer to reduce the case base while maintaining the prediction accuracy of the system, it is better to use DCBM model than ACCM applied only during acquisition. Once analysed DCBM model alone, we test the combination between acquisition (ACCM) and learning (DCBM) stages at the same time.

Table 2 shows (from column 10 to 15) the results of such combination. In this case, the ACCM final case base will be the initial one for the DCBM policies. Before examining this question in detail, let us notice that there are two results to highlight: the percentage of cases maintained by our DCBM policies and the final case base size when finishing both processes. The percentage of cases maintained during oblivion (**obliv**) is computed using this formula $\frac{\#finalcases}{\#finalcasesACCM} \times 100$, which shows the behavior of the DCBM policies. The percentage of final case base size (**size**) shows the percentage of case base maintained from the original training set, it is computed using this formula $\frac{\#finalcases}{\#traincases} \times 100$.

Table 2. Results for all methods using an update parameter KNN = 5 Av1 shows the mean value for all datasets. We use paired t-test at the level of 5% significance, where a • and a ∘ stand for a significant improvement or degradation of DCBM policies and ACCM to 1NN

Ref	cbr		cbm		cbr		cbr			cbm			cbm	
	1NN	*size*	**ACCM**	**size**	**OL**	*size*	**OCE**	*size*	**OL**	*obliv*	*size*	**OCE**	*obliv*	*size*
BL	76.15	100.0	77.27	•97.44	78.73	•88.69	78.73	•88.69	78.11	85.89	83.69	79.04	•90.25	87.94
BC	95.86	100.0	95.43	77.36	95.99	67.93	95.99	97.61	**96.26**	38.79	30.01	95.98	96.67	74.78
HC	73.36	100.0	70.91	∘86.14	**81.24**	•88.79	**81.24**	•88.79	**81.24**	•81.49	70.19	80.16	•88.53	76.26
CA	81.76	100.0	82.19	84.30	82.63	89.40	82.63	89.40	82.47	86.80	73.17	82.47	87.35	73.63
MA	63.93	100.0	64.53	89.19	62.11	51.44	63.39	77.77	55.88	∘ 7.55	6.73	**64.91**	78.95	70.42
TG	96.13	100.0	96.13	95.87	**96.66**	97.44	**96.66**	97.44	63.92	∘ 0.23	0.22	96.60	97.72	93.69
HH	72.82	100.0	72.12	85.63	75.56	•87.86	75.56	•87.86	75.19	14.38	12.32	**76.23**	•88.12	75.47
HS	74.07	100.0	75.55	79.67	74.81	86.74	74.81	86.74	75.18	29.28	23.33	**77.03**	85.27	67.94
HP	77.99	100.0	77.33	87.67	78.58	87.67	78.58	87.67	74.75	47.83	41.93	77.87	86.09	75.48
IO	86.92	100.0	87.20	83.79	87.74	91.45	87.74	91.45	**88.01**	54.25	45.45	87.74	91.16	76.38
IR	95.33	100.0	**96.66**	89.03	95.33	97.03	95.33	97.03	91.33	∘ 8.56	7.63	96.00	97.60	86.96
LB	83.38	100.0	83.04	77.38	**87.04**	88.50	**87.04**	87.91	81.14	52.14	40.35	86.47	86.65	67.05
SY	82.15	100.0	83.83	•78.38	87.15	•92.09	**87.28**	•91.65	86.22	•87.96	68.94	86.22	•88.58	69.43
VE	69.43	100.0	68.13	72.36	69.53	80.33	69.53	80.33	68.36	67.40	48.77	68.38	72.23	52.27
VT	86.65	100.0	90.78	•79.23	92.60	•95.47	92.60	•95.47	91.96	•40.39	32.00	**92.86**	•95.03	75.30
Av	81.06	100.0	81.40	84.22	**83.04**	86.05	**83.14**	89.72	79.33	46.82	38.98	**83.19**	88.68	74.86

We concentrate on different observations in table 2 that allow us to express points in favour of the DCBM model.

- Reduction of the case base during the acquisition stage is not enough. As the results show in ACCM column and we have also noticed previously, it is necessary to delete "harmful" knowledge during the problem solving process.
- The DCBM using the problem solving process helps the system to obtain a more accurate and reduced case base.
- The reduction obtained using DCBM augment the prediction accuracy of standard 1NN algorithm, with the exception of the combination between ACCM and OL. The combination does not work because it is too much aggressive with the case base, as expected previously when defined.
- On the other hand, OL works properly if it is not combined with ACCM, even though it has a great reduction policy to select the cases for being removed from the case base. In conclusion, OL can be only applied alone.
- The combination of ACCM with OCE does not improve often the performance of OCE applied alone. However, the combination has a higher reduction than OCE alone and also improves on average previous prediction accuracy.

5 Conclusions

This paper proposes a model for case base maintenance that uses the dynamics of the problem solving process to search for the optimal case base while maintaining the prediction accuracy. The experimental study demonstrates that the DCBM model using different policies manage to get the initial objectives: it optimizes the case base while it improves on average the prediction accuracy of the system. Our further work will be focused on testing the model in recommender systems in order to analyze a dynamic environment with our dynamic model. We also think of testing different case reduction methods on acquisition stage.

Acknowledgements. This work is supported by the *Ministerio de Ciencia y Tecnologia,* Grant No. TIC2002-04160-C02-02. We wish to thank *Enginyeria i Arquitectura La Salle - Ramon Llull University* for their support to our Research Group.

References

[Golobardes et al., 2002] Golobardes, E., Llorà, X., Salamó, M., and Martí, J. (2002). Computer Aided Diagnosis with Case-Based Reasoning and Genetic Algorithms. *Knowledge-Based Systems,* (15):45–52.

[Harmon, 1996] Harmon, M. (1996). Reinforcement learning: A tutorial.

[Leake and Wilson, 2000] Leake, D. and Wilson, D. (2000). Remembering Why to Remember: Performance-Guided Case-Base Maintenance. In *Proceedings of the Fifth European Workshop on Case-Based Reasoning,* pages 161–172.

[Markovitch, S. and Scott, P.D., 1988] Markovitch, S. and Scott, P.D. (1988). The Role of Forgetting in Learning. In *Proceedings of the Fifth International Conference on Machine Learning,* pages 459–465.

[Merz and Murphy, 1998] Merz, C. J. and Murphy, P. M. (1998). UCI Repository for Machine Learning Data-Bases [http://www.ics.uci.edu/~mlearn/MLRepository.html]. *Irvine, CA: University of California, Department of Information and Computer Science.*

[Minton, 1985] Minton, S. (1985). Selectively generalizing plans for problem solving. In *Ninth International Joint Conference on Artificial Intelligence,* pages 596–599. Morgan Kaufmann.

[Pawlak, 1982] Pawlak, Z. (1982). Rough Sets. In *International Journal of Information and Computer Science,* volume 11.

[Portinale et al., 1999] Portinale, L., Torasso, P., and Tavano, P. (1999). Speed-up, quality and competence in multi-modal reasoning. In *Proceedings of the Third International Conference on Case-Based Reasoning,* pages 303–317.

[Salamó and Golobardes, 2003]	Salamó, M. and Golobardes, E. (2003). Hybrid Deletion Policies for Case Base Maintenance. In *Proc. of the sixteenth International FLAIRS Conference,* pages 150–154. AAAI Press.
[Scott, 1983]	Scott, P. (1983). Learning: The construction of a posteriori knowledge structures. In *Proceedings of the Third National Conference on Artificial Intelligence.*
[Smyth and Keane, 1995]	Smyth, B. and Keane, M. (1995). Remembering to forget: A competence-preserving case deletion policy for case-based reasoning systems. In *Proceedings of the Thirteen International Joint Conference on Artificial Intelligence,* pages 377–382.
[Smyth and Mckenna, 2001]	Smyth, B. and Mckenna, E. (2001). Competence Models and the maintenance problem. *Computational Intelligence,* 17(2):235–249.
[Sutton and Barto, 1998]	Sutton, R. and Barto, A. (1998). *Reinforcement Learning. An introduction.* The MIT Press.

A Case Base Seeding for Case-Based Planning Systems

Flavio Tonidandel and Márcio Rillo

Centro Universitário da FEI – UniFEI,
Av. Humberto de A. Castelo Branco, 3972,
09850-901 - São Bernardo do Campo - SP - Brazil
flaviot@fei.edu.br, rillo@lsi.usp.br

Abstract. This paper describes a Case Base Seeding system (CBS) that can be used to seed a case base with some random cases in order to provide minimal conditions for the empirical tests of a Case-Based Planning System (CBP). Random case bases are necessary to guarantee that the results of the tests are not manipulated. Although these kind of case bases are important, there are no references about CBS systems in the literature even from those CBP systems that claim to use some similar systems. Therefore, this paper tries to overcome this deficiency by modeling and implementing a complete random Case Base Seeding process.

1 Introduction

There are many case-based planning (CBP) systems in the literature [4],[5],[7],[12],[13]. A CBP system is known as a planner that retrieves potential cases from a case base that can become solutions to problems. Any case-based planner needs a case base with suitable cases that allows the system to work properly and to have some advantages over generative planning systems.

Differently of the CBP literature, Case Base Seeding systems are rare in the literature, although they are important for CBP system. In fact, most CBP systems seed their case bases in order to perform their tests. However, these seeding processes are either composed by hand or they are specific for some domains. In addition, there are no available explanations about them in the literature.

This lack of a generic and random seeding process in the literature can become an obstacle to CBP systems researches to perform their empirical tests and, consequently, to analyze their systems performance suitably.

In this paper, a Case Base Seeding system is presented by focusing on how to create random and suitable states in planning domains. The seeding system creates a random problem by creating consistent initial and goal states. This random problem is then solved by a generative planner, called FF [3], which produces a solution (a plan), and consequently, a new case for the case base.

This paper is a detailed extension of the CBS System used in [12] and it is structured as follow: in the section 2, some case bases used by some CBP systems are described. In section 3, the main part of the CBS system is detailed followed by the section 4 that describes the complete CBS system. Section 5 discusses some issues of CBS system and, finally, section 6 concludes this paper.

2 Case Bases in CBP Systems

A Case-Based Planning system is a planner that uses past experience, stored in cases, to solve the problems. Cases in a CBP system are previous plans or traces of planning solutions. In order to be an efficient system, a Case-Based Planner must use a case base with a large variety of cases. This is because the performance of a CBP system depends on the cases in the case base since the adaptation of a similar case can be computational expensive in general. Therefore, to retrieve a very similar case or even a complete-solution case improves the performance of the CBP system by decreasing the time spent by the adaptation phase. Obviously, anyone can design and create, by hand, a case base with many very similar cases to some specific problems in order to increase the performance of the CBP system.

However, to perform any empirical test in CBP systems, a case base with random cases must be available. This random case base would avoid any kind of manipulation that permits a system to work with many very similar cases and that consequently turns the tests not completely valid. The problem is that there is no schema or a formal process to create case bases with random cases available in the literature.

Some CBP systems use pre-defined or given examples to fill their case bases up or to perform their tests, e.g., DerSNLP+EBL [4]. Others systems like CAPER [5] and Caplan/CBC [7] use automatically creation of cases, but they need either a start pivot (Caplan/CBC) or were designed for some specific domains (CAPER – UM Translog Domain).

The Prodigy/Analogy [13] is another CBP system that uses an automatic creation process of cases. This process creates random problems and, consequently, random cases. However, it is specific for some domains, like logistic transportation, one-way-rocket and machine-shop domains, and Veloso [13] does not let available any detail of how this seeding system really works.

To summarize, the cases creation processes used by CBP systems are either domain specific or no specification of how these processes work can be found. In other words, there is no generic process that researches can use to create random cases for CBP systems.

This paper intends to overcome this deficiency by defining a Case Base Seeding system that can be used for any Case-Based Planner to create their own random case base. It is a detailed and extended version of the seeding system presented and used in the Far-Off system [12].

3 Creating Random Consistent States

A Case Base Seeding system can be designed by using a generative planning system to produce sound plan-solutions and, consequently, suitable cases. However, this generative planner just performs its task if an initial state and a goal state are available. In fact, the most important piece of a CBS system is a process that creates correct and consistent states in the application's domain. A state in a planning domain is a set of instantiated predicates [12].

In order to generate random and consistent states in a certain domain, some additional information must be added by the user. Usually, the available information for a planning system is only the domain and the problem features both in PDDL language [6]. The domain features describe the actions and the predicates, besides the types of the elements that the predicates can handle. The problem features describe the initial state, the Goal State and the elements for each type in the domain.

Each predicate can handle some specific types of elements, which are the possible values for the predicate's free variables. A predicate can become a fact when its all free variables are instantiated, i.e., a fact is a grounded predicate.

Definition 1 (A Fact). *A fact is a grounded (instantiated) predicate.*

Definition 2 (The Predicate of a Fact). *The predicate of a fact is the not-grounded format of the fact.*

However, these information about domain, predicates, actions and problem are not enough for a CBS system, i.e., it is not possible to create consistent states with this information only.

Therefore, additional information, containing the semantic of each predicate and their relations with others predicates, is necessary. For example, in the Blocks World domain, this additional information must define that *holding* and *handempty* predicates cannot appear together in a state. In fact, this additional knowledge is difficult to be obtained from the information described in PDDL language.

This additional information, from now on simply denominated domain semantic, encapsulates semantic features of the domain in terms of predicates relations that can not be extracted from actions, but only from consistent states features.

To describe the relations among predicates, some definitions are stated. These definitions must encapsulate negative and positive interactions:

Definition 3 (Positive Existence). *The Positive Existence of a predicate p is the set of predicates and facts that must be in the state where p is true.*

Definition 4 (Negative Existence). *The Negative Existence of a predicate p is the set of predicates and facts that can not be in the state where p is true.*

Definition 5 (Positive Absence). *The Positive Absence of a predicate p is the set of predicates and facts that must be in the state where p is not true.*

Definition 6 (Negative Absence). *The negative absence of a predicate p is a set of predicates and facts that can not be in a state where p is not true.*

In order to clarify the use and the importance of the definitions above, the Blocks World domain with 4 blocks *(A,B,C* and *D)* can be considered. In this domain, the facts *on(A,C), holding(A)* and *clear(B)* are the predicates that are in the negative existence of the fact *on(A,B).* It means that the facts *on(A,C), holding(A)* and *clear(B)* can not be together, with the fact *on(A,B)* in the same state. In fact, not only *on(A,C)* can not exists in the same state of *on(A,B),* but also any predicate of the form *on(A,x).*

The seeding system permits a generic specification of negative and positive existence and absence. For example, for a *on(x,y)* predicate, the following predicates can not be in the same state: *holding(x), clear(y)* and *on(x,_).* The others definitions encapsulate the complementary three possibilities of Existence and Absence relations of a specific predicate. There are some facts, however, that must be in all states of the domain. They are called fixed facts:

Definition 7 (Fixed Facts). *The Fixed Facts is a set of facts that must be true in all states of the domain.*

In a typed domain, some predicates restrict their variables to some values. An example can be illustrated in Logistic domain. The predicate *at(x,y)* means that *x* is at *y*, where *x* is a *package* or a *vehicle* and *y* must be a *location* like *city* or *airport.* However, the semantic of this predicate in a domain is restricted to: *at(airplane,airport)* and *at(package,y),* i.e, when the *x* variable is instantiated with an *airplane,* the *location* denoted by *y* must necessarily be an *airport.* On the other hand, the type *package* does not require any kind of restriction for the *y* variable. This information is not explicit in the domain's definition. To describe such feature the following definition is stated:

Definition 8 (Restricted Facts). *The Restricted Facts is a set of facts of a specific predicate that can really exist in a certain domain.*

The purpose of the above definition is to restrict the facts that can be created between the predicate definition and all elements encapsulated in each type. These restricted facts must be provided by the user because there is no information about it in problem or domain features. They can also be automatically extracted from the domain specification through some state invariant extractor, like TIM [1].

Some predicates are even more restricted. They can appear in a state one time at most and others can appear one time at least. This feature is then defined:

Definition 9 (At-Least-One Predicate). *The At-Least-One predicate is that predicate which facts can be true in a state at least one time.*

Definition 10 (At-Most-One Predicate). *The At-Most-One predicate is that predicate which a fact can be true in a state at most one time.*

With all above definitions, a process to create a consistent state is defined in Figure. 1. As stated before, a process to create a consistent state is necessary to define a process that creates a case base. In order to produce a consistent state, a set of all facts of the domain is necessary.

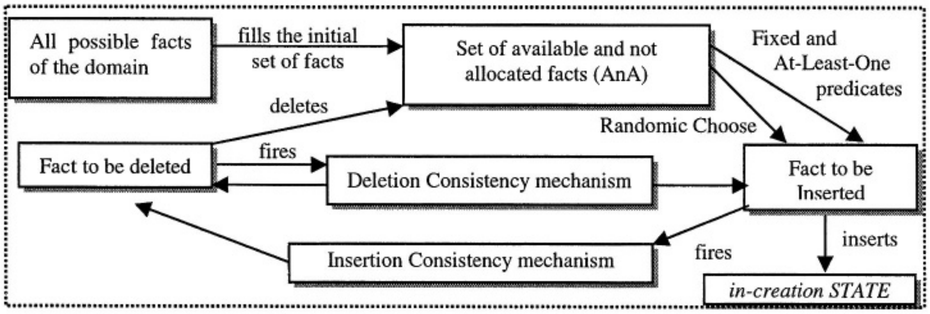

All possible facts of the domain
fills the initial set of facts
Set of available and not allocated facts (AnA)
Fixed and At-Least-One predicates
deletes
Randomic Choose
Fact to be deleted
fires
Deletion Consistency mechanism
Fact to be Inserted
inserts
fires
Insertion Consistency mechanism
in-creation STATE

Fig. 1. The diagram of the consistent state creation process

The process to create this set, called *Available and not Allocate* predicates set (AnA), follows an initial seed example and the restrictions configured by the user.

This initial seed is an example of a problem, including an initial state and all the elements of each type of the domain. This information can also be given by the user, like the Prodigy/Analogy system [13].

The most important contribution of the seed example is the determination of the values that can instantiate a fixed predicate. Since the fixed facts are defined as those predicates that must be in all states, they will also be in the initial state of the seed example. Therefore, the possible facts of each fixed predicate can be easily extracted from the seed example.

Besides the number and specification of the elements and the determination of fixed facts, the seed example is also necessary to provide important information about the consistency between the initial state and the goal state. Some predicates accept only some values for their variables, and these values are defined in an example of a possible initial state. For example, in the logistic domain, the predicate *at(truck,city)* can not be instantiated with any *truck* and any *city,* because some *trucks* are specific of some *cities* and they can not travel to other *cities.* So, the CBS system must consider this information.

In order to complete the restriction specified in definition 8, an specific and special variable must be used to extract information from the initial state of the seed problem example. This variable, called *InSt,* restricts the values of an specific variable to those presented in the seed example.

Observe that the seed example avoids the user to specify all the restricted, fixed facts, and all elements of the domain by hand. Therefore, the process to create the AnA set follows the rules specified below:

- ♦ Extract from the seed example all elements of each type
- ♦ For each predicate, all possible combination of elements in their variables are generated considering:
  - ♦ Each element must instantiate one and only one variable of the predicate, eliminating any fact that has repeated element in their variables.
  - ♦ If the predicate is a fixed fact, then consider only the facts that exist in the seed example.
  - ♦ If an element will instantiate a variable labeled as *InSt,* then consider only the facts that match any other fact in the seed example.

During the AnA set creation, all facts that are grounded At-Least-One predicates compose a specific sub-set and all facts that configure as Fixed Fact compose another specific sub-set.

The second step is to choose one fact from AnA to compose the *in-creation* state. This choosing process follows a heuristic that chooses the most restricted predicates first. These restricted predicates are those classified as Fixed and At-Least-One predicates. This is because both predicates must be in all states and they can probably prune other facts.

The process chooses the Fixed Facts first and then selects At-Least-One predicates randomly. Each selection fires the Consistency Mechanism that applies one of their rules. When all Fixed and any different At-Least-One predicates from the chosen ones are not available in AnA set, the process starts the generic random process that can choose any fact present in AnA.

The CBS system uses a set of consistency rules that guarantee the creation of a consistent state (*in-creation* State). The consistency rules are allocated in two Consistency Mechanisms: Insertion and Deletion Consistency Mechanisms.

Definition 11 (Rules of the Consistency Mechanism).
In Insertion Consistency Mechanism, there are three rules: For each insertion of a fact f in in-creation State, where f is a grounded predicate of the general predicate p:

- ***Consistency Rule 1 (ICM-R1):***
  All facts that fit in Positive Existence predicates of p are included in the in-creation State and deleted from AnA set.
- ***Consistency Rule 2 (ICM-R2):***
  All facts that fit in Negative Existence predicates of p are deleted from AnA set.
- ***Consistency Rule 3 (ICM-R3):***
  If the predicate p is configured as The-Most-One predicate, all facts of predicate p are deleted from AnA

In the same way, the Deletion Consistency Mechanism has two rules: For each fact f deleted from AnA, where f is a grounded predicate of the general predicate p:

- ***Consistency Rule 1 (DCM-R1):***
  Each fact that fits in the Negative Absence of p must be also deleted from AnA.
- ***Consistency Rule 2 (DCM-R2):***
  Each fact that fits in the Positive Absence of p must be inserted in in-creation State.and deleted from AnA.

Therefore, the CBS system performs a repetition of insertion and the application of the Consistency rules. When the AnA becomes empty, the process stops and the *in-creation State* becomes a potential consistent state following the user configuration.

During the application of the consistent rules, it is possible to detect that fail occurred and consistent state can not be created. For example, the rule DCM-R2 can detect there is no predicate that it must insert. Therefore, if any rule fails to accomplish their tasks no consistent state is guaranteed to be created.

The process of deletion and insertion performed by the consistent rules are fired when a fact is deleted from AnA or inserted in the *in-creation* State. Since this process is not cyclic, it can not falls in a infinite and recursive looping caused by a wrong configuration, which would probably accuse fail.

4 Creating a Random Case Base

With a random state creation process defined, the entire process of a case base creation can be designed. As mentioned before, a case is made by a plan generated by a planner. This planner will be the generative planner called Fast-Forward (FF) [3] that is a heuristic search planner. The FF system uses a heuristic (FF-heuristic) to perform its planning task: to find a plan from a given initial state to another state where the goal is satisfied.

4.1 Creating a Random Initial and Goal States

The process to create random and consistent states, described before, is used to create the initial and the goal states. Although the initial state is a complete and consistent state that is straightforward released by the state creation process, the goal state is not. In fact, the difference between initial state and the Goal State is that the latter can not be a complete state. In theory, a goal state may be a complete state, however, as verified in many available planning problems for different domains, only some predicates are allowed to constitute a Goal State. For example, in the Blocks World domain, only the predicate *on(x,y)* is relevant to the goal. The same observation can be made for *at(x,y)* predicate in Logistic domain and the *served(x)* in Miconic domain.

Therefore, to let the Goal State similar to those presented in the problems, it is defined the relevant predicates:

Definition 12 (Relevant Predicates). *The relevant Predicates are those predicates that can be part of a goal state.*

The relevant predicates of a domain must be configured by the user or extract from a seed example. The goal state has another restriction: the number of the relevant predicates is also random. For example, in the Blocks World domain with 10 blocks, a goal can be only a composition of two *on(X,Y)* predicates, like *on(A,B)* and *on(B,C),* and not a composition of all possible and consistent combination of *on(x,y)* predicate.

Therefore, the process to create a goal state can be more than the straightforward use of a complete random state. It requires two filters: The Relevant Predicates Filter and the Random Number of Facts Filter. These filters are applied in a random and consistent state, called first-stage goal state, created by the process described above.

Both filters prune the predicates in the Goal State. The Relevant Predicate Filter will delete any fact from the first-stage goal state that is not a relevant predicate, creating the second-stage goal state with only relevant predicates.

The second filter, the Random Number of Facts Filter, is now applied. Its first step is to choose randomly the number of facts, denominated Gn, which will be left in the state. This number has the range from 1 to the number of facts in the second stage state. The second step is to choose randomly Gn facts of the second stage, creating the third (the final stage) of the goal state.

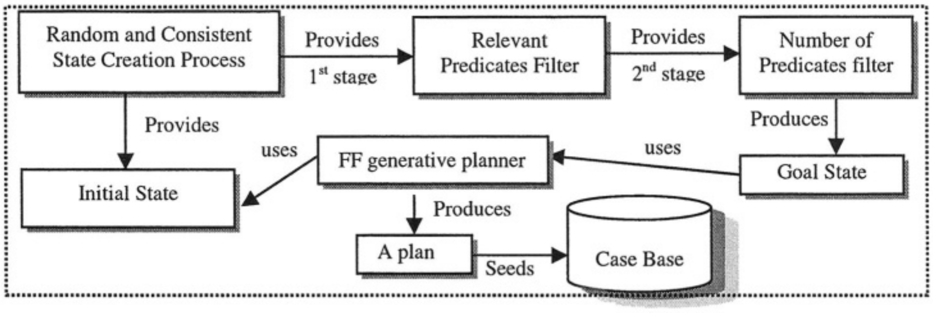


Fig. 2. The complete model of the Case Base Seeding Process

The final stage of the goal state is then created by Gn relevant facts from a consistent and complete state resulted by the creation process.

4.2 Producing a Plan

With an initial and goal states established, the process to produce a plan is about to start. However, a last verification must be necessary. Until now, consistent initial and goal states are created, but nothing can guarantee that both states are consistent with each other. As a last verification, the FF-Heuristic is used to estimate a number of action between the initial and goal states. The FF-heuristic is very useful for many case base planning purposes [10], because it is a good estimation of the possible number of actions between a state and the goal.

In order to guarantee the consistency between the initial and goal states, the FF-heuristic result, applied for both states, must be a finite number. If the estimation result is infinite, then there is no possible solution between those states. In this case, the process restarts and others consistent states are generated as initial and goal states. Otherwise, if the result estimation is a finite number, then the FF planner is fired to produce a plan between initial and goal states. This plan is transformed to a new case.

A generative planner is necessary because cases are composed by plans or traces of them. Given a sound plan as a solution for a problem, a case can be created. The Figure 2 summarizes the case base creation process in a more general view.

4.3 Maintenance Process for Case Bases

As a post-seeding process, a maintenance policy must be applied in the case base in order to let it representative and not redundant. For that, any maintenance policy found in the literature, like min-injury [11], Type-based [9] and case-addition policy

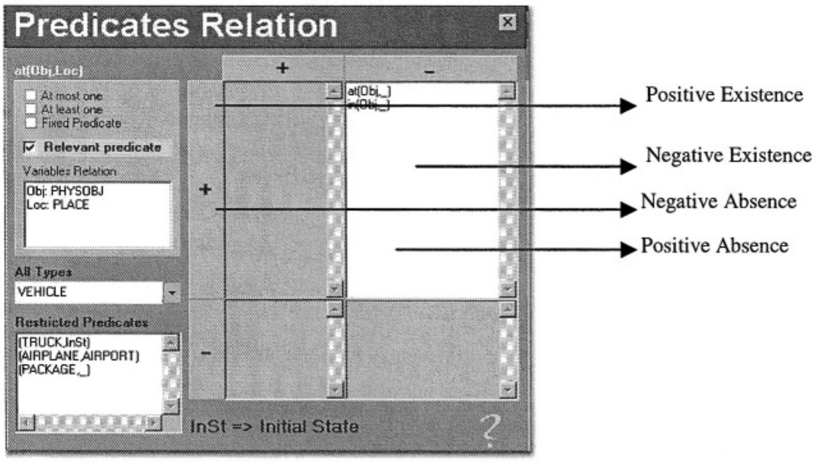


Fig. 3. Example of the *at(obj,Loc)* predicate configuration in the CBS

[14] can be applied to the case base in order to improve its competence. The competence of a case is the range of problems that it can solve [8].

The maintenance policies will reduce the redundant cases and others cases that do not contribute to increase the competence, but only to create ‘hot spots’ – spots of concentration of cases. After the use of any maintenance process, a short case base will be created, keeping its original competence and an uniform distribution of cases.

If the short case base becomes smaller than necessary, the seeding process can be called to produce more cases to fill the case base up again. This will create a cycle of seeding and maintenance process that will increase the quality and the representativeness of the case base.

5 Empirical Tests and Discussion

The Case Base Seeding process presented in this paper was implemented and used in the Far-Off system [12]. The figure 3 shows the configuration window of the CBS seeding used by the Far-Off system.

In the Far-Off system empirical tests, this CBS was used to create case bases over more than 3000 cases for each problem and domain. The creation of those case bases allowed diversified tests in many STRIPS domains, as showed in [12]. It is important to highlight that the CBS described in this paper concerns about STRIPS-model domains. Therefore, the CBS system is not suitable to describe the semantics of all existed planning domains. This is because the definitions in this paper were extracted only from STRIPS domains, and most of them were in planning competition.

The CBS system can be improved to work with more complex domains that handle numerical parameters and resources. In addition, it can be also improved by using a learning system that can learn and extract information from states, problems and domain defined in PDDL language. An algorithm, called DISCOPLAN [2] discovers state constraints (invariants) of a domain automatically. It uses this information to improve the planning efficiency. It can discover, for example, some information about type constraint, like those defined in definition 8, or also extract some relation as those defined from definition 3 to definition 6. In the future, the CBS system can use DICOPLAN or TIM [1] to extract information automatically.

6 Conclusion

This paper describes a Case Base Seeding system (CBS) that can be used to construct a case base with random cases. It is suitable for empirical tests of Case-Based Planning Systems (CBP).

Seeding systems are not easily found in the literature, although many CBP systems use some random case bases in their tests. This paper just describes a complete and generic CBS system that can be applied in STRIPS-model domains.

The CBS system can be improved in the future, by incorporating more complex features and by using some automatic processes to extract states constraints and invariant automatically.

References

1. Fox, M.; Long, D. The Automatic Inference of State Invariants in TIM. *Journal of Artificial Intelligence Research,* 9 ,1998. 367-421.
2. Gerevini A.; Schubert, L. Discovering State Constraints in DISCOPLAN: Some New Results. In: Proc. of AAAI-2000. AAAI Press, 2000.
3. Hoffmann, J.; Nebel, B.. The FF Planning System: Fast Plan Generation Through Heuristic Search. *Journal of Artificial Intelligence Research.* 14, 2001. 253 – 302.
4. Ihrig, L.H.; Kambhampati,S. Storing and Indexing Plan Derivations through Explanation-based Analysis of Retrieval Failures. *Journal of Artificial Intelligence Research,* 1 , 1991. 1-15
5. Kettler, B. P. *Case-Based Planning with a High-Performance Parallel Memory.* 1995. PhD Thesis, University of Maryland Park, Maryland, 1995.
6. Long, D.; Fox, M. *The 3rd International Planning Competition - IPC'2002.* Available in <http://www.dur.ac.uk/d.p.long/competition.html>.
7. Munõz-Avila, H. ; Weberskirch, F. Planning for Manufacturing Workpieces by Storing, Indexing and Replaying Planning Decisions. In: Proc. of AIPS-96, AAAI Press, 1996.
8. Smyth, B., McKenna, E. Building Compact Competent Case-Bases. In: Proc. of. ICCBR'99. Althouff, K., Bergmann, R., Branting, K. (Eds.), Lecture Notes in Artificial Intelligence, v. 1650. Springer-Verlag, 1999.
9. Smyth, B., Keane, M. Remembering to Forget: A Competence-preserving Case-deletion Policy for Case-based Reasoning Systems. In: Proc. of IJCAI'95, 14., Morgan Kaufmann, 1995.
10. Tonidandel, F.; Rillo, M. An Accurate Adaptation-Guided Similarity Metric for Case-Based Planning In: Proc. of ICCBR-2001. Aha, D., Watson, I. (Eds.), Lecture Notes in Artificial Intelligence. v.2080, Springer-Verlag, 2001.
11. Tonidandel, F.; Rillo, M. Releasing Memory Space through a Case-deletion policy with a Lower bound for Residual Competence. In: In: Proc. of ICCBR-2001. Aha, D., Watson, I. (Eds.), Lecture Notes in Artificial Intelligence, v.2080, Springer-Verlag, 2001.
12. Tonidandel, F.; Rillo, M. The Far-Off system: A Heuristic Search Case-Based Planning. In: Proc. of aips'02, AAAI Press, 2002.
13. Veloso, M. Planning and Learning by Analogical Reasoning. *Lecture Notes in Artificial Intelligence,* v.886. Springer-Verlag, 1994.
14. Zhu J., Yang Q. Remembering to Add: Competence-preserving Case-Addition Policies for Case-Base Maintenance. In: Proc. of IJCAI'99. M.Kaufmann, 1999.

Handling Numeric Criteria in Relaxed Planning Graphs

Oscar Sapena[1] and Eva Onaindía[2]

[1] Depto. de Ciencias de la Computación e Inteligencia Artificial, Universidad de Alicante, Spain
osapena@dccia.ua.es

[2] Depto. de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Spain
onandia@dsic.upv.es

Abstract. Nowadays, one of the main techniques used in heuristic planning is the generation of a relaxed planning graph, based on a *Graphplan*-like expansion. Planners like *FF* or *MIPS* use this type of graphs in order to compute distance-based heuristics during the planning process. This paper presents a new approach to extend the functionality of these graphs in order to manage numeric optimization criteria (problem metric), instead of only plan length optimization. This extension leads to more informed relaxed plans, without increasing significantly the computational cost. Planners that use the relaxed plans for further refinements can take advantage of this additional information to compute better quality plans.

1 Introduction

The problem of domain-independent planning is a very complex problem. In fact, the problem is *PSPACE-complete* even if some severe restrictions are applied [2]. Nowadays, one of the best techniques to deal with these complex problems is heuristic planning. Planners like *FF* [8], *LPG* [6], *VHPOP* [12] or *MIPS* [4], use different heuristic functions to guide the search, and all of them have demonstrated a very competitive performance.

The general principle for deriving heuristics is to formulate a simplified (or relaxed) version of the problem. Solving the relaxed problem is, in general, easier than solving the original problem. The solution of the relaxed problem is used as heuristic to estimate the distance to the goal. One of the most common relaxations is to ignore the negative effects (or delete list in *STRIPS* notation) of the actions. This idea was first proposed by *McDermott* [9] and, since then, a big number of planners have adopted this technique (i.e., *HSP* [1], *GRT* [10], *Sapa* [3], *FF,* etc.)

Nowadays, the planning community is working on extending the functionality of their planners to deal with more expressive problems. Many real-world problems have several characteristics that can hardly be expressed in pure propositional *STRIPS:* complex action conditions, numeric functions, durative actions, uncertainty, etc. Most of these features can be modeled with the *PDDL 2.1* [5] language, but planners must be adapted to support the new extensions. One of the main contributions of *PDDL 2.1* is the possibility of specifying an optimization criterion for a planning problem. This criterion, called problem metric, consists of a numeric expression, which has to be maximized or minimized. Therefore, the user can ask the planner to optimize, for example, the fuel consumption in a transportation problem, rather than the commonly used criteria as the plan length or plan duration.

In this paper we propose a new technique to extend the relaxed planning graphs. This technique allows to deal with numeric optimization criteria. In spite of there are heuristic planners that can handle problem metrics (like *Metric-FF* [7] or *MIPS* [4]), their heuristic functions are still based on the plan length of the relaxed problem solution. This paper shows how this technique can be applied to improve the quality of the heuristic functions and, consequently, the quality of the final plans.

2 The Relaxed Planning Graph

The relaxed planning graph (*RPG*) is a graph, based on a *Graphplan*-like expansion where delete effects are ignored. In this section we describe the traditional generation of a *RPG,* although taking into account the numeric part of *PDDL 2.1.* Firstly, we have to formalize some concepts. A (numeric) planning problem is defined as a tuple $\langle F, P, A, I, G, M \rangle$, where:

- F is a set of numeric variables, called fluents.
- P is a set of logical propositions.
- A is the set of actions.
- I is the initial state.
- G is a set of goals.
- M is the problem metric (numeric optimization criterion).

The numeric extensions of *PDDL 2.1* imply that, in addition to the propositions P, we have a set F of numeric variables (fluents). A state s is thus defined as a set of propositions $p(s)$ and a set of rational numbers $v(s)$ that represent the values for each fluent in that state:

$$s = \langle p(s), v(s) \rangle / p(s) \subset P \wedge v(s) = \{v_1, \dots, v_n\} : value(f_i, s) = v_i, \forall f_i \in F$$

An *expression* is an arithmetic expression over F and the rational numbers, using the operators +, -, * and /. The value of an expression *exp* in a state s is represented as $value(exp, s)$. A *numeric constraint* is a triple $\langle exp, comp, exp' \rangle$ where *exp* and exp' are expressions, and $comp \in \{<, \leq, =, \geq, >\}$ is a comparator. A *numeric effect* is a triple $\langle f_i, ass, exp \rangle$ where $f_i \in F$ is a fluent, $ass \in \{:=, + =, - =, * =, / =\}$ is an assignment operator, and *exp* is an expression. The outcome of applying a numeric effect in a state s, written $nresult(\langle f_i, ass, exp \rangle, s)$, is another state in which the value of fluent f_i has been modified with the value of *exp,* using the assignment operator *ass*.

```
$t = 0; P_0 = s_0$                                  // Initialization
while $G \not\subseteq P_t$ do                      // New expansion stage
    $A_t = \{a \in A / pprec(a) \subseteq P_t\}$    // Action level
    $P_{t+1} = P_t \bigcup add(a), \forall a \in A$ // Proposition level
    if $P_{t+1} = P_t$ then fail endif
    $t = t + 1$
endwhile
```

Fig. 1. Traditional *RPG* expansion

Actions in this numeric framework can have numeric preconditions and effects. Therefore, the preconditions of an action $a \in A$ can be propositional, $pprec(a) \subset P$, or numeric constraints, $nprec(a)$. Likewise, the effects of a can be propositional, $add(a)$ and $del(a)$ for positive and negative effects respectively, or numeric effects, $neff(a)$. Regarding the problem goals, we restrict ourselves to propositional goals ($G \subset P$) for simplicity. This is the same simplification that the *Sapa* planner [3] does. However, the techniques presented in this paper can be easily translated to other approaches (like *Metric-FF* [7], which only ignores decreasing numeric effects). Finally, the problem metric (M) is an expression which value must be minimized. A maximization problem can be turned into a minimization problem just multiplying the metric expression by -1. Now, we can describe a relaxed planning problem, which ignores all delete effects of all actions.

Definition 1. *Assuming a planning problem $P_p = \langle F, P, A, I, G, M \rangle$, the relaxation a^+ of an action $a \in A$, $a = \langle pprec(a), nprec(a), add(a), del(a), neff(a) \rangle$, is defined as:*

$$a^+ = \langle pprec(a), \emptyset, add(a), \emptyset, \emptyset \rangle$$

The relaxed planning problem is $P_p^+ = \langle F, P, A^+, I, G, M \rangle$, where $A^+ = \{a^+ / a \in A\}$.

The first step to compute a solution for a relaxed planning problem is to build the *RPG*, The traditional *RPG* building algorithm generates proposition and action levels alternately. The first level (P_0) is a proposition level which contains all the propositions that are true in the starting state (s_0). Action levels contain all actions that are applicable in the previous level, and the following proposition levels are extended with the add effects of these actions. The expansion of the *RPG* finishes when a proposition level containing all top-level goals is reached, or when it is not possible to apply any new action. Figure 1 shows this process.

3 Handling Optimization Criteria

The basic idea to take into account the optimization criterion in the *RPG* stage is to include some information about the actions cost according to the problem metric. Nevertheless, considering all possible different situations which can arise

```
$prog\_prop = s_0$                                              // Initialization
$cost(p) = \begin{cases} 0 & \text{, if } p \in s_0 \\ \infty & \text{, otherwise} \end{cases}, \forall p \in P$
while $\exists g \in G / cost(g) = \infty$ do                   // New expansion stage
    if $prog\_prop = \emptyset$ then fail endif
    $c = min(cost(p)), \forall p \in prog\_prop$                 // Level cost
    $P_c = \{p / p \in prog\_prop \wedge cost(p) = c\}$          // Proposition level
    $prog\_prop = prog\_prop - P_c$
    $A_c = \{a \in A / cost(p) \leq c, \forall p \in pprec(a)\}$ // Action level
    for all $a \in A_c$ do                                       // Programming action effects
        $cost\_reach(a) = \sum cost(p), \forall p \in pprec(a)$
        for all $p \in add(a) \wedge (cost\_reach(a) + cost(a) < cost(p))$ do
            $prog\_prop = prog\_prop \cup \{p\}$
            $cost(p) = cost\_reach(a) + cost(a)$
        endfor
    endfor
endwhile
```

Fig. 2. Proposed *RPG* for problem metric optimization

due to modifications in the fluent values is unfeasible. Therefore, our proposal consists in evaluating (an estimate of) the cost of the actions in the current state (s_0). This simplification is completely acceptable in problems where the costs and the consumption of resources are not very dependent on the state in which the actions are applied. Most of the planning problems fulfill this requirement. For example, the fuel consumption of a plane highly depends on static information like the flight distance. State-dependent information, like the number of passengers in a particular flight, only affects slightly.

The main difference with respect to the traditional *RPG* is that the levels of the graph do not represent time steps, but costs according to the problem metric (M). Thus, to compute the levels we have to estimate the cost of applying an action a:

$$\begin{array}{l} cost(a) = value(M, nresult(neff(a), s_0)) - value(M, s_0) + \varepsilon \\ \texttt{if } cost(a) < \varepsilon \texttt{ then } cost(a) = \varepsilon \texttt{ endif} \end{array} \tag{1}$$

The cost of an action a is computed as the increase in the metric value caused by the application of a in s_0 (we apply the numeric effects of a to s_0, regardless of whether the preconditions of a hold in s_0 or not). The small ε included in the cost represents that every action, even those that do not affect the metric value, have a cost. This way, if there is no metric defined in the problem, our *RPG* is equivalent to the traditional one. Finally, we check the cost to be positive. We ignore the action effects which decrease the metric value, since it will cause an out of order expansion of the graph. Then, these actions are considered to have the minimum cost. The algorithm for the *RPG* expansion can be formalized as figure 2 shows.

The algorithm uses the list *prog_prop* (programmed propositions) in order to store the propositions which will be inserted in the graph. Each programmed

Table 1. Domain description of the example problem

operator	param.	pprec	add	del	neff
Load	?c	at T ?c ∧ at P ?c	in P T	at P ?c	∅
Unload	?c	at T ?c ∧ in P T	at P ?c	in P T	∅
Drive	?c1 ?c2	at T ?c1	at T ?c2	at T ?c1	driven+=distance ?c1 ?c2

proposition p has an associated cost $cost(p)$. Initially, the *prog_prop* list only contains the propositions that hold in the current state s_0. These propositions have no cost since they are currently true. The rest of the propositions are not achieved yet and, therefore, have an infinite cost.

The graph expansion starts with the generation of the first propositional level. The propositional levels (P_c) are indexed through the cost of their propositions, so all propositions in a level will have the same cost. The level cost (c) is computed as the minimum cost of the programmed propositions, in order to build the graph from lower to higher cost values. The respective action level (A_c) contains the actions which preconditions have a cost c or lower. The positive effects of these actions will be added to the *prog_prop* list only if they have not been achieved before with a lower cost. Let's suppose that a is the action that produces a proposition p; the cost of p $(cost(p))$ is computed as the addition of:

- The cost of achieving a $(cost\_reach(a))$: this cost is computed as the sum of the a preconditions costs.
- The cost of applying a $(cost(a))$, defined in (1).

The *RPG* expansion finishes when all top-level goals are achieved, or when the *prog_prop* list becomes empty. If the *prog_prop* list is empty, then no new action can be applied and, therefore, the goals cannot be achieved.

This algorithm can be used to improve the heuristic information extracted from the relaxed plans. Section 3.1 shows an example to compare the traditional *RPG* with our proposal. Moreover, the computational complexity of the algorithm is polynomial since the traditional one is proved to be polynomial [8], and the additional calculations (action and proposition costs) can be done in polynomial time.

3.1 Example of the *RPG* Expansion

We will illustrate the *RPG* expansion through a *Logistics*-like example problem (see figure 3). In this problem, there are three cities (C_1, C_2 and C_3), one truck (*T*) and one package (*P*). The truck and the package are initially located in C_1 and C_2 respectively. The distance between the cities is shown in figure 3 through the labels in the roads. The goal is to carry the package to city C_3, minimizing the driven distance (the problem metric is: *minimize*(*driven*)). Table 1 summarizes the domain description.

The *RPG* for this problem is shown in table 2. This *RPG* has more levels than the traditional one, but the number of actions and propositions per level is lower.

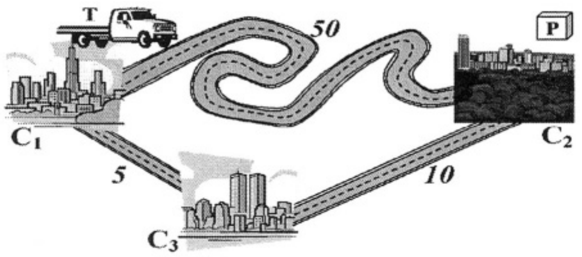


Fig. 3. Initial state in the logistics example problem

Table 2. Relaxed planning graph for the example problem

P_0	A_0	$P_{5+\varepsilon}$	$A_{5+\varepsilon}$	$P_{15+2\varepsilon}$	$A_{15+2\varepsilon}$
at T C_1 *at P* C_2	*Drive* C_1 C_2 *Drive* C_1 C_3	*at T* C_3	*Drive* C_3 C_1 *Drive* C_3 C_2	*at T* C_2	*Drive* C_2 C_1 *Drive* C_2 C_3 *Load* C_2
$P_{15+3\varepsilon}$	$A_{15+3\varepsilon}$	$P_{15+4\varepsilon}$	$A_{15+4\varepsilon}$	$P_{20+5\varepsilon}$	
in P T	*Unload* C_1 *Unload* C_2 *Unload* C_3	*at P* C_1	*Load* C_1	$\underline{\textit{at P } C_3}$	

The first action level (A_0) contains the actions that are directly executable: drive from C_1 to C_2 and C_3. The cost of having truck T in C_3 is 5 (since this is the distance between C_1 and C_3), so the action effects (*at T* C_3) will be programmed with the cost $5 + \varepsilon$. Actions which do not affect the metric, like the load and unload operations, have a cost of ε. The expansion finishes when the goal (*at P* C_3) is achieved.

The relaxed plan obtained using our *RPG* is the following:

$C_3\ C_2 \longrightarrow$ *Load* $C_2 \longrightarrow$ *Unload* C_3

The following plan has been computed by means of the *FF's* relaxed plan extraction algorithm [7]:

$P2 = \{\textit{Drive } C_1\ C_2, \textit{Drive } C_1\ C_3\} \longrightarrow \textit{Load } C_2 \longrightarrow \textit{Unload } C_3$

The benefits of our proposal can be easily observed just comparing both relaxed plans. It can be observed that $P1$ is almost executable (it only needs one action to go from C_2 to C_3). Plan $P2$, however, has hard conflicts since actions *Drive* C_1 C_2 and *Drive* C_1 C_3 are mutually exclusive. Moreover, action *Drive* C_1 C_2 should not be included in the plan because of its high cost. Obviously, if the planner only takes into account the relaxed plan length to compute its heuristics, this technique does not bring many advantages. On the contrary, planners that use the information provided by relaxed plan will find a valuable help building the final plan.

4 Results

The *RPG* expansion for handling numeric criteria has been implemented in the *SimPlanner v3* planner [11]. *SimPlanner v3* is a heuristic planner that can take advantage of the presented improvements on the relaxed planning graphs. In other heuristic planners that only use the relaxed plan length, like *FF* or *HSP,* the extra valuable information from the *RPG* is not fully exploited.

Our proposed expansion and the traditional one are compared in this section. Both techniques are implemented in the same planner (*SimPlanner v3*). This way, the results are not influenced by other characteristics of the planner. For this reason, we have not included comparisons between *SimPlanner v3* and other heuristic planners, since the results would not provide reliable information about the performance of our proposal.

The domains used in this comparison are numeric domains introduced in the third international planning competition (IPC'02). The domains used in the competition are described at

`http://planning.cis.strath.ac.uk/competition.`

Table 3 shows the results obtained in this comparison for the problems in *DriverLog, ZenoTravel* and *Depots* numeric domains. The *DriverLog* is a variation of *Logistics* where trucks need drivers. Drivers can move along different road links than trucks. The optimization criterion in. *DriverLog* is to minimize an instance-specific linear combination of total time, driven distance and walked distance. *ZenoTravel* is a transportation domain, where objects must be transported via aeroplanes. The optimization criterion is to minimize an instance-specific linear combination of total time and fuel consumption. The *Depots* domain is a combination of *Logistics* and *Blockworld* domains, where some blocks must be transported with trucks between depots and arranged in a certain order. The optimization criterion is to minimize the overall fuel consumption.

Results on table 3 show the plan quality of the solutions, according to the problem metric defined, and the running times. These results show that our proposal improves the plan quality in most of the problems, and solves more problems than using the traditional approach. On average, the computed plans are 1.87, 1.25 and 1.23 times better in the *DriverLog, ZenoTravel* and *Depots* domains respectively. However, due to the heuristic nature of *SimPlanner v3*, there are a few problems where the traditional approach obtains better plans.

Regarding the running times, table 3 shows that our proposal takes more time than the traditional one. This is mainly due to three factors:

– **The Application of Formula (1) to Estimate the Actions Costs.** However, in these domains this computation has only a slight effect on the running time since the costs are static, that is, they do not depend on the state. And, for the same reason, the estimated costs for these domains are only computed once in the planning process.

Table 3. Comparison between the results obtained with the proposed expansion and with the traditional one (in the form *proposed/traditional*) for the numeric *DriverLog*, *ZenoTravel* and *Depots* domains. Quality depends on the problem metric (greater numbers stand for more costly plans), and time is measured in seconds

	DriverLog		ZenoTravel		Depots	
Prob.	Quality	Time	Quality	Time	Quality	Time
1	777/777	0.01/0.01	13564/13564	0.01/0.01	32/42	0.01/0.01
2	999/1625	0.08/0.13	6786/6786	0.01/0.01	43/53	0.04/0,03
3	1406/1406	0.03/0.03	6758/6758	0.01/0.01	29/29	0.22/0.15
4	1119/986	0.05/0.05	27000/27000	0.01/0.01	64/50	0.54/0.44
5	1056/1270	0.05/0.05	3978/3978	0.02/0.02	80/259	0.92/2.54
6	2095/2466	0.03/0.02	25097/25097	0.03/0.03	313/313	205.2/195.5
7	1876/1476	0.07/0.04	11198/11198	0.02/0.02	37/57	0.1/0.1
8	3418/3472	0.17/0.09	29677/55480	0.04/0.04	43/43	0.43/0.43
9	4091/9144	1.43/1.53	13275/12644	0.14/0.06	231/409	88.47/79.92
10	241/3211	0.07/0.26	177368/175218	0.36/0.2	27/27	0.27/0.27
11	753/738	0.18/0.07	25505/65093	0.15/0.05	229/196	8.35/13.05
12	6713/5346	5.86/0.7	52206/38547	0.2/0.08	299/389	34.09/27.54
13	2886/3556	0.83/0.76	112468/95657	0.55/0.22	27/27	1.93/2
14	11153/-	2.22/-	430417/183591	3.51/0.27	43/43	2.4/2.42
15	3561/3464	2.79/0.59	59835/205701	1.92/1.65	237/214	28.21/34.35
16	16829/39771	110.6/24	64119/87530	4.14/1.48	31/32	0.29/0.27
17	20655/93031	60/27.7	190845/384645	36.94/12.36	29/31	0.53/0.53
18	70917/82266	283.4/91.3	67610/71944	20.42/7.26	108/127	37.4/36.73
19	-/-	-/-	235378/257104	44.46/15.2	48/48	4.65/4.7
20	11555/27884	462/70.7	111671/355711	26.91/20.21	217/-	28.46/-
	1.87 better	4.26 slower	1.25 better	2.36 slower	1.23 better	1.03 slower

- **The Number of Graph Levels.** This number is from 10 to 70 times greater in our proposal, although the number of actions per level is lower.
- **The Number of Actions in the Graph.** This number is often greater than in the traditional approach due to the expansion of many unuseful low-cost actions in first place. In the *DriverLog* domain, for example, a lot of unnecessary *walk* actions are inserted in the graph. This is the main reason for the greater running times in most of the problems.

These three factors slightly increase the time consumed in the *RPG* creation. However, the overall time increment is more significant as *SimPlanner v3* has to build many *RPGs* when solving a problem. For example, the number of created *RPGs* is greater than 10^5 for some problems.

5 Conclusions and Future Work

In this paper, we have presented an extension to the traditional relaxed planning graph generation. A relaxed planning graph is based on a GraphPlan-like expansion, where delete effects are ignored. The use of these graphs is widely used in many heuristic planners.

The proposed extension allows to take into account the optimization criterion (or metric) of the problem. During the graph expansion, an estimate of the cost of the actions is computed according to the problem metric. This estimate is used to expand the less costly actions in first place. Practical results show that our proposal, in general, obtains better solutions than the traditional approach.

The relaxed planning graphs are the starting point for the heuristic estimators of many planners. Therefore, all the improvements on these graphs can help the planners increase their performance. *Metric-FF* [7], for example, proposes an extension to deal with the numeric effects of the actions. The *Metric-FF* expansion is completely compatible with our proposed expansion and, therefore, both techniques could be implemented in the same planner. However, there are several features that have not been addressed in the relaxed planning graph framework yet. Handling probabilities and sensing actions, for example, could allow the planners to face problems with uncertainty more efficiently.

Acknowledgements

This work has been partially funded by projects *SAMAP TIC2002-04146-C05-04, MCyT DPI2001-2094-C03-03* and *UPV 20020681.*

References

1. B. Bonet and H. Geffner, 'Planning as heuristic search', *Artificial Intelligence Journal,* **129,** 5–33, (2001).
2. T. Bylander, 'The computational complexity of propositional STRIPS planning', *Artificial Intelligence,* **69,** 165–204, (1994).
3. M.B. Do and S. Kambhampati, 'Sapa: A multi-objective metric temporal planner', *Journal of Artificial Intelligence Research,* **20,** 155–194, (2003).
4. S. Edelkamp, 'Taming numbers and durations in the model checking integrated planning system', *Journal of Artificial Intelligence Research,* **20,** 195–238, (2003).
5. M. Fox and L. Derek, 'PDDL2.1: An extension to PDDL for expressing temporal planning domains', *Journal of Artificial Intelligence Research,* **20,** 61–124, (2003).
6. A. Gerevini, A. Saetti, and I. Serina, 'Planning through stochastic local search and temporal action graphs in LPG', *Journal of Artificial Intelligence Research,* **20,** 239-290, (2003).
7. J. Hoffmann, 'The Metric-FF planning system: Translating 'ignoring delete lists' to numeric state variables', *Journal of Artificial Intelligence Research,* **20,** 291–341, (2003).
8. J. Hoffmann and B. Nebel, 'The FF planning system: Fast planning generation through heuristic search', *Journal of Artificial Intelligence Research,* **14,** 253–302, (2001).
9. D. McDermott, 'A heuristic estimator for means-ends analysis in planning', *Proceedings of the International Conference on Artificial Intelligence Planning Systems,* **3**, 142–149, (1996).

10. I. Refanidis and I. Vlahavas, 'The GRT planning system: Backward heuristic construction in forward state-space planning', *Journal of Artificial Intelligence Research,* **14,** 115–161, (2001).
11. O. Sapena, E. Onaindia, M. Mellado, C. Correcher, and V. Vendrell, 'Reactive planning simulation in dynamic environments with VirtualRobot', *Innovations in Applied Artificial Intelligence, LNCS,* **3029,** 699–707, (2004).
12. H.L.S. Younes and R.G. Simmons, 'VHPOP: Versatile heuristic partial order planner', *Journal of Artificial Intelligence Research,* **20,** 405–430, (2003).

Constrainedness and Redundancy by Constraint Ordering*

Miguel A. Salido[1] and Federico Barber[2]

[1] Dpto. Ciencias de la Computación e Inteligencia Artificial, Universidad de Alicante, Campus de San Vicente, Ap. de Correos: 99, E-03080, Alicante, Spain
[2] Dpto. Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Camino de Vera s/n, 46071, Valencia, Spain
{msalido, fbarber}@dsic.upv.es

Abstract. In constraint satisfaction, a general rule is to tackle the hardest part of a search problem first. In this paper, we introduce a parameter (τ) that measures the constrainedness of a search problem. This parameter represents the probability of a problem being feasible. A value of $\tau = 0$ corresponds to an over-constrained problem and no states are expected to be solutions. A value of $\tau = 1$ corresponds to an under-constrained problem and every state is a solution. This parameter can also be used in heuristics to guide search. To achieve this parameter, a simple random or systematic sampling is carried out to compute the tightnesses of each constraint. New heuristics are developed to classify the constraints from the tightest constraint to the loosest constraint and to remove redundant constraints in constraint satisfaction problems. These heuristics may accelerate the search due to inconsistencies can be found earlier and the absence of such redundant constraints eliminate unnecessary checking and save storage space.

Keywords: Constraint Satisfaction Problems, constrainedness, heuristics.

1 Introduction

Many real problems in Artificial Intelligence (AI) as well as in other areas of computer science and engineering can be efficiently modeled as Constraint Satisfaction Problems (CSPs) and solved using constraint programming techniques. Some examples of such problems include: spatial and temporal planning, qualitative and symbolic reasoning, diagnosis, decision support, scheduling, hardware design and verification, real-time systems and robot planning.

These problems may be soluble or insoluble, they may be hard or easy. How to solve these problems have been the subject of intensive study in recent years.

* This work has been partially supported by the grant DPI2001-2094-C03-03 from the Spanish Government.

Some works are focused on the constrainedness of search. Heuristics of making a choice that minimises the constrainedness can reduce search [4].The constrainedness "knife-edge" measures the constrainedness of a problem during search [12].

Furthermore, some other works try to reduce a CSP by identifying and removing redundant constraints [10] since a constraint that allows all the possible value assignments of the variables on which it is defined, none of the constraints tuples has to be checked. Thus, the absence of such a constraint eliminates unnecessary checking and saves storage space [7]. However, identifying redundant constraint is hard, in general [10].

However, most of the work is focused on general methods for solving CSPs. They include backtracking-based search algorithms. While the worst-case complexity of backtrack search is exponential, several heuristics to reduce its average-case complexity have been proposed in the literature [3]. For instance, some algorithms incorporate features such as ordering heuristics. Thus, some heuristics based on *variable ordering* and *value ordering* [8] have been developed, due to the additivity of the variables and values. However, constraints are also considered to be *additive,* that is, the order of imposition of constraints does not matter; all that matters is that the conjunction of constraints be satisfied [1]. In spite of the additivity of constraints, only some works have be done on constraint ordering heuristic mainly for arc-consistency algorithms [11, 5].

Here, we introduce a parameter that measures the "constrainedness" of the problem. This parameter called τ represents the probability of a problem being feasible and identify the tightnesses of constraints. This parameter can also be applied in heuristics to guide search. To achieve this parameter, we compute the tightnesses of each constraint. Using this tightnesses, we have developed two heuristics to accelerate the search. These heuristics perform a constraint ordering and redundant constraints removal. They can easily be applied to any backtracking-based search algorithm.

The first one classifies the constraints by means of the tightnesses, so that the tightest constraints are studied first. This is based on the principle that, in goods ordering, domain values are removed as quickly as possible. This idea was first stated by Waltz [13] " *The base heuristic for speeding up the program is to eliminate as many possibilities as early as possible"* (p. 60). An appropriate ordering is straightforward if the constrainedness is known in advance. However in the general case, a good classification is suitable to tackle the hardest part of the search problem first.

The second heuristic is based on the idea of reducing a CSP into an "easy problem" by removing redundant constraints [10].

In the following section, we formally define a CSP and summarize some heuristics. In section 3, we define a well-known definition of constrainedness of search problems. A new parameter to measure the constrainedness of a problem is developed in section 4. In section 5, we present our constraint ordering heuristic. Section 6 summarizes the conclusions and future work.

2 Definitions and Heuristics

In this section, we review some basic definitions as well as heuristics for constraint ordering, constrainedness and redundant constraints removing for CSPs.

2.1 Definitions

Definition 1. A constraint satisfaction problem (CSP) consists of:

- a set of variables $V = \{v_1, v_2, ..., v_n\}$
- each variable $v_i \in V$ has a set D_{v_i} of possible values (its domain). We denote d_{v_i} the length of domain D_{v_i}.
- a finite collection of constraints $C = \{c_1, c_2, ..., c_k\}$ restricting the values that the variables can simultaneously take.

Definition 2. A solution to a CSP is an assignment of values to all the variables so that all constraints are satisfied.

Definition 3. A redundant constraint is a constraint that can be removed without changing the solutions.

There is not a broadly accepted definition of constrainedness. We adopt the following definition of constrainedness:

Definition 4. The constrainedness of a problem is a predictor of computational cost to find a solution.

2.2 Heuristics

The experiments and analyses by several researchers have shown that the ordering in which variables and values are assigned during the search may have substantial impact on the complexity of the search space explored.In spite of the additivity of constraints, only some works have be done on constraint ordering.

Wallace and Freuder initiated a systematic study to identify factors that determine the efficiency of constraint propagation that achieve arc-consistency [11]. Gent et al. proposed a new constraint ordering heuristic in AC3, where the set of choices is composed by the arcs in the current set maintained by AC3 [5]. They considered the remaining subproblem to have the same set of variables as the original problem, but with only those arcs still remaining in the set.

On the other hand, many other problems may contain redundant constraints. Edward Tsang in [10] proposed the possibility of reducing a CSP to an "easy problem" by removing redundant constraints. However, redundancy is in general difficult to detect; but some redundant constraints are easier to identify than others. In [2], which focuses on binary CSPs, a number of concepts for helping to identify redundant binary constraints are introduced. For example, a constraint in a binary CSP can be removed if it is path-redundant (definition 3-19 [10]). The time complexity of path-redundant is $O(nd^3)$. However, since not every problem can be reduced to an easier problem, one should judge the likelihood of succeeding in reducing the problem in order to justify the complexity of procedures such as path-redundant.

Heuristics of making a choice that minimises the constrainedness of the resulting subproblem can reduce search over standards heuristics [4]. Walsh studied the constrainedness "knife-edge" in which he measured the constrainedness of a problem during search in several different domains [12]. He observed a constrainedness "knife-edge" in which critically constrained problems tend to remain critically constrained. This knife-edge is predicted by a theoretical lower-bound calculation. Many of these algorithms focus their approximate theories on just two factors: the size of the problems and the expected number of solutions which is difficult to obtain.

In [4], Gent et al. present a parameter that measures the constrainedness of an ensemble of combinatorial problems. They assume that each problem in an ensemble has a state space S with $|S|$ elements and a number, Sol of these states are solutions. Any point in the state space can be represented by a N-bit binary vector where $N = log_2(|S|)$. Let $\langle Sol \rangle$ be the expected number of solutions averaged over the ensemble. They defined constrainedness, κ, of an ensemble by,

$$\kappa =_{def} 1 - \frac{log_2(\langle Sol \rangle)}{N} \tag{1}$$

However, this parameter defines the constrainedness of constraint satisfaction problems in general, but not of an individual problem.

3 Constrainedness τ

In this section, we introduce a parameter called τ that measures the constrainedness of the problem. This parameter represents the probability of a problem being feasible. This parameter lies in the range [0, 1]. A value of $\tau = 0$ corresponds to an over-constrained and no state is expected to be a solution ($\langle Sol \rangle = 0$). A value of $\tau = 1$ corresponds to an under-constrained and every state is expected to be a solution ($\langle Sol \rangle = \prod_{v \in V} d_v$). This parameter can also be used in a heuristic to guide search. To this end, we take advantage of the tightnesses of each constraint to classifying them from the tightest constraint to the loosest constraint and to removing some redundant constraints. Thus, a search algorithm can tackle the hardest part of the problem first with a lower number of constraints.

As we pointed out, a simple random or systematic sampling is performed to compute τ, where there is a target population (states), and a sampled population is composed by $s(n)$ random and well distributed states where s is a polynomial function.

As in statistic, the user selects the desired precision by the size of the sample $s(n)$. We study how many states $st_i : st_i \leq s(n)$ satisfy each constraint c_i (see Figure 1). Thus, each constraint c_i is labeled with p_{c_i}: $c_i(p_{c_i})$, where $p_{c_i} = st_i/s(n)$ represents the proportion of possible states, that is, the tightnesses of the constraint.

In this way, given the set of probabilities $\{p_{c_1}, ..., p_{c_k}\}$, the number of solutions can be computed as:

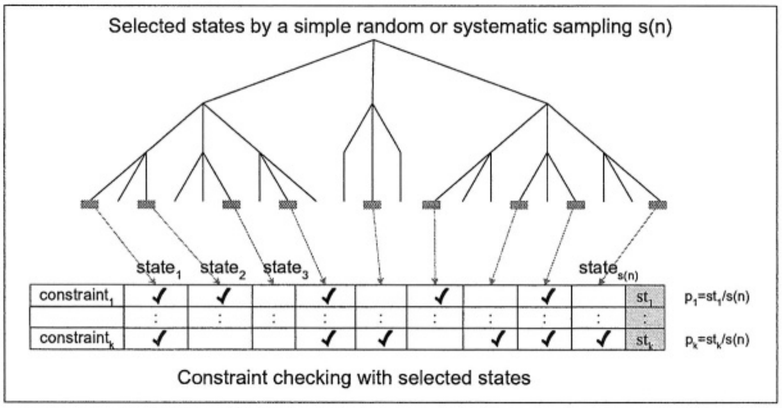

Selected states by a simple random or systematic sampling s(n)
state1
state2
state3
state s(n)
constraint1
constraintk
st1
stk
p1=st1/s(n)
pk=stk/s(n)
Constraint checking with selected states

Fig. 1. From non-ordered constraint to ordered constraint

$$\langle Sol \rangle := (\prod_{v \in V} d_v) \times (\prod_{c_i \in C} (p_{c_i})) \tag{2}$$

This equation is equivalent to the obtained in [4]. However, our definition of constrainedness is given by the following equation:

$$\tau := \prod_{c_i \in C} (p_{c_i}) \tag{3}$$

τ is a parameter that measures the probability that a randomly selected state is a solution, that is, the probability this state satisfies the first constraint (p_{c_1}), the second constraint (p_{c_2}) and so forth, the probability this state satisfies the last constraint (p_{c_k}). Thus, this parameter lies in the range [0, 1] that represent the constrainedness of the problem.

We present the pseudo-code of computing τ.

```
Computing the constrainedness $\tau$

Inputs: A set of $n$ variables, $v_1, ..., v_n$;
        For each $v_i$, a set $D_i$ of possible values (the domain)
        A set of constraints, $c_1, ..., c_k$.
Outputs: The constrainedness $\tau$.
1.- From the number of states generated by the Cartesian product of the variable
domain bounds, a random and well distributed sample with $s(n)$ states is selected.
2.- With the selected sample of states $s(n)$, we compute how many states $st_i : st_i \leq$
$s(n)$ satisfy each constraint $c_i, i = 1..k$. Thus, $c_i$ is labelled with $p_{c_i} = st_i/s(n)$.
3.- $\tau := \prod_{c_i \in C}(p_{c_i})$
```

4 Some Heuristics Using τ

To compute τ, it is necessary to obtain the tightnesses of each constraint, represented by the following set of probabilities $\{p_{c_1}, ..., p_{c_k}\}$. We can take advantage of this information to develops some heuristics to guide search or to improve efficiency.

The first heuristic is committed to classify the constraints so that a search algorithm can manage the hardest part of a problem first. Figure 2 shows the constraints in the natural order and classified by tightnesses. If the tightest constraints are very constrained ($\tau \approx 0$), the problem may be over-constrained. However, if these tightest constraints are under-constrained ($\tau \approx 1$) then, the problem will be under-constrained.

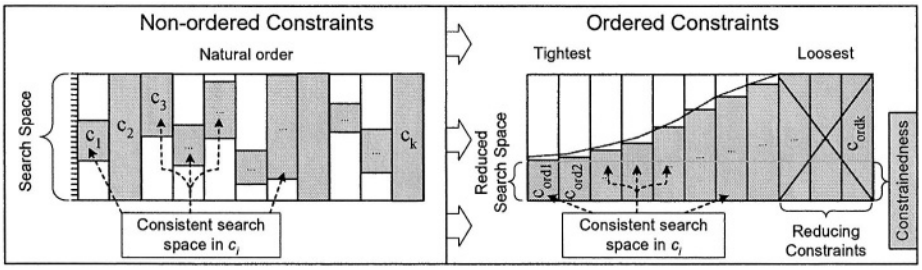


Fig. 2. From non-ordered constraints to ordered constraints: Constrainedness

The second heuristic is focused on constraint removing in random CSPs. The loosest constraints are analyzed and the redundant ones are removed of the problem.

Let's see these two heuristics.

4.1 Constraint Ordering Heuristic

This easy heuristic takes advantage of τ making use of the probabilities of constraints $p_{c_1}, p_{c_2}, ..., p_{c_k}$. This heuristic classifies the constraints in ascending order of the labels p_{c_i} so that the tightest constraints are classified first $p_{c_{ord1}}, p_{c_{ord2}}, ..., p_{c_{ordk}}$ (see Figure 2).

Thus, a backtracking-based search algorithm can tackle the hardest part of a search problem first and inconsistencies can be found earlier and the number of constraint checks can significantly be reduced.

4.2 Redundant Constraint Removing Heuristic

In many random problems, some constraints may be redundant. A redundant constraint is a constraint that can be removed without changing the solutions. Some redundant constraint may be easier to identify than others [10].

In global constraints, to improve consistency a symbolic reasoning based on rewriting and redundant constraint introduction may help CSP solvers to find the result more directly [6].

However, in general, redundant constraints do not always save search effort [7]. For example, if a constraint allows all the possible value assignments of the variables on which it is defined, none of the constraints tuples has to be checked. The absence of such a constraint, in fact, eliminates unnecessary checking and saves storage space [7].

We will focus on this line, where our main goal is to identify redundant constraints in order to significantly reduce the number of constraint checks. Anyway, in case of global constraint, this heuristic may help to identify the set of redundant constraints.

We can identify two different types of redundant constraints:

- Constraints that are made redundant by any other single constraints (e.g., $x + y < 10$ is made redundant by $x + y < 5$).
- Constraints that due to their topology satisfy all possible assignments (e.g., $x + y < 5$ with domains $x : \{1, 2\}$ and $y : \{1, 2\}$).

The first type of constraints is not directly related to τ since the tightnesses p_{c_i} of each constraint c_i is not so relevant to identify redundancy.

The second type of constraints may be identified by the tightnesses of constraints. Constraint with $p_c = 1$ may be redundant due to all the random selected states in the sample satisfy this constraint. The coincidence can be given that all the selected states are consistent and however not to be a redundant constraint. So, we identify some types of constraints that can be removed if $p_c = 1$ and they satisfy a simple formula.

The main type of constraints is arithmetic constraints of the form:

$$\sum_{i=1}^{n} \alpha_i v_i \leq \gamma \tag{4}$$

Each constraint c_i in the form (6) with $p_{c_i} = 1$ can be eliminated if:

$$\sum_{i=1}^{n} \beta_i v_i \leq \gamma : \begin{cases} \beta_i = D_i^+ & \alpha_i > 0 \\ \beta_i = 0 & \alpha_i = 0 \\ \beta_i = D_i^- & \alpha_i < 0 \end{cases} \tag{5}$$

where D^- and D^+ correspond to the lower and upper variable domain bound.

In this way, all constraints with $p_c = 1$ satisfying the above formula can be removed. The absence of such constraints eliminate unnecessary checking and save storage space [7].

5 Evaluation of τ and Heuristics

In this section, we evaluate our parameter τ and both heuristics. To estimate the constrainedness of random problems we compare τ with the actual constrained-

Table 1. Random instances $< n, c, d >$, n:variables, c:constraints and d:domain size

Problems	*actual constrainedness*	*Parameter* τ	*Number of States*	*Number of Solutions*	*Number of Estimated Sol.*	*% Error*
$< 3, 5, 5 >$	0.09	0.07	125	11.2	8.7	2%
$< 3, 5, 10 >$	0.05	0.043	1000	50	43	0.7%
$< 3, 10, 5 >$	0.024	0.013	125	3	1.6	1.12%
$< 5, 5, 5 >$	0.04	0.038	3125	125	118.7	0.2%
$< 5, 10, 5 >$	0.008	0.01	3125	25	31.2	0.19%
$< 5, 10, 10 >$	0.0045	0.0034	100000	453	340	0.1%

ness by obtaining all solutions of random problems. To evaluate the constraint ordering heuristic we incorporated our heuristic to well-known CSP solvers: Backtracking (BT), Generate&Test (GT), Forward Checking (FC) and Real Full Look Ahead (RFLA)[1], because they are the most appropriate techniques for observing the number of constraint checks. Finally, we evaluated the redundant constraint removing heuristic over random problem using BT.

5.1 Evaluating τ

In our empirical evaluation, each random CSP was defined by the 3-tuple: $< n, c, d >$,where n was the number of variables, c the number of constraints and d the domain size. The problems were randomly generated by modifying these parameters. We evaluated 100 test cases for each type of problem. We present the average actual constrainedness by obtaining all solutions, our estimator τ choosing a sample of $s(n) = 7n^2$ states, the number of possible states, the average number of possible solutions, the average number of estimate solutions using τ and the error percentage.

Table 1 shows some types of random problems. For example in problems with 5 variables, each with 5 possible values and 5 constraints $< 5, 5, 5 >$, the number of possible states is $d^n = 5^5 = 3125$, the average number of solutions is 125, so the actual constrainedness is 0.04. With a sample of $7n^2 = 175$ states, we obtain an average number of 6.64 solutions. Thus, our parameter $\tau = 0.038$ and the number of estimate solutions of the entire problem is 118.7. In this way, the error percentage is only 0.2%.

5.2 Evaluating the Constraint Ordering Heuristic

The n-queens problem is a classical search problem to analyse the behaviour of algorithms. Table 2 shows the amount of constraint check saving in the n-queens problem.

[1] BT, GT, FC and RFLA were obtained from CON'FLEX. It can be found in: http://www-bia.inra.fr/T/conflex/ Logiciels/adressesConflex.html

Table 2. Constraint check saving using GT , BT, FC and RFLA in the n-queens problem

	GT&GT+CO	BT&BT+CO	FC&FC+CO	RFLA&RFLA+CO
queens	*Constraint Check Saving*	*Constraint Check Saving*	*Constraint Check Saving*	*Constraint Check Saving*
5	2.1×10^4	2.4×10^2	150	110
10	4.1×10^{11}	3.9×10^7	1.4×10^5	9.3×10^4
20	1.9×10^{26}	3.6×10^{18}	9.6×10^{14}	6.03×10^{11}
50	2.4×10^{70}	3.6×10^{52}	3.1×10^{44}	1.6×10^{32}
100	2.1×10^{143}	2.1×10^{106}	4.5×10^{93}	1.8×10^{66}
150	5.2×10^{219}	3.7×10^{161}	6.8×10^{142}	2.1×10^{100}
200	9.4×10^{295}	8.7×10^{219}	9.9×10^{198}	2.2×10^{134}

We incorporated our constraint ordering (CO) to well-known CSP solver: GT+CO, BT+CO, FC+CO and RFLA+CO. Here, the objective is to find all solutions. The results show that the amount of constraint check saving was significant in GT+CO and BT+CO and lower but significant in FC+CO and RFLA+CO. This is due to these techniques are more powerful than BT and GT.

Table 3. The redundant constraint removing heuristic in random instances $< n, c, d >$

Problems	*Redundant constraints*	*Constraint checks (entire problem)*	*Constraint checks (filtered problem)*	*Constraint checks saving*
$< 5,5,5 >$	1.75	15625	10325	5300
$< 5,10,5 >$	3.5	31250	20312	10938
$< 5,20,5 >$	7	62500	40625	21875
$< 5,5,10 >$	1.75	5×10^5	3.3×10^5	1.7×10^5
$< 5,10,10 >$	3.5	1×10^6	6.5×10^5	3.5×10^5
$< 5,20,10 >$	7	2×10^6	1.3×10^6	7×10^5

5.3 Evaluating the Redundant Constraint Removing Heuristic

We evaluated this heuristic over random problems as presented in section 5.1. In this case, all constraints are global constraints, that is, all constraints have arity n. Table 3 shows for each type of constraints, the average number of redundant constraints (red_c), the amount of constraint checks for the entire problem, the amount of constraint checks for the filtered problem (without redundant constraints) and the amount of constraint checks saving using backtracking (BT). It can be observed that the amount of constraint checks for the entire problem was $d^n c$. The constraint checks for the filtered problem was $d^n(c - red_c)$. So, the constraint check saving was $d^n(red_c)$, that corresponds a saving of 35%. These formulas may be standardized for backtracking. If the average number of redundant constraints is known, the constraint check saving can be obtained by $d^n(red_c)$.

6 Conclusion and Future Work

In this paper, we introduce a parameter (τ) that measures the "constrainedness" of a search problem.τ represents the probability of a problem being feasible. A value of $\tau = 0$ corresponds to an over-constrained problem. $\tau = 1$ corresponds to an under-constrained problem. This parameter can also be used in a heuristic to guide search. To achieve this parameter, we compute the tightnesses of each constraint. We can take advantage of this tightnesses to classify the constraints from the tightest constraint to the loosest constraint and to remove redundant constraints. Using the constraint ordering heuristic and the redundant constraint removing heuristic, the search can be accelerated due to inconsistencies can be found earlier and the number of constraint checks can significantly be reduced.

Furthermore, these heuristic techniques are appropriate to solve problems as a distributed CSPs [9] in which agents are committed to solve their subproblems.

For future work, we are working on integrating constraint ordering with variable ordering in centralized and distributed CSPs.

References

1. R. Bartk, 'Constraint programming: In pursuit of the holy grail', *in Proceedings of WDS99 (invited lecture), Prague, June,* (1999).
2. A. Dechter and R. Dechter, 'Removing redundancies in constraint networks', *Proceeding of National Conference on Artificial Intelligence (AAAI),* 105–109, (1987).
3. R. Dechter and J. Pearl, 'Network-based heuristics for constraint satisfaction problems', *Artificial Intelligence,* **34**, 1–38, (1988).
4. I.P. Gent, E. MacIntyre, P. Prosser, and T Walsh, 'The constrainedness of search', *In Proceedings of AAAI-96,* 246–252, (1996).
5. I.P. Gent, E. MacIntyre, P. Prosser, and T Walsh, 'The constrainedness of arc consistency', *Principles and Practice of Constraint Programming,* 327–340, (1997).
6. A. Liret, P. Roy, and F. Pachet, 'Constraints satisfaction and symbolic reasoning for reactive control systems', *CP Workshop on Constraints in Control,* (1999).
7. M. Sabin and E.C. Freuder, 'Detecting and resolving inconsistency and redundancy in conditional constraint satisfaction problems', *Configuration Papers from the AAAI Workshop, WS-99-05,* (1999).
8. N. Sadeh and M.S. Fox, 'Variable and value ordering heuristics for activity-based jobshop scheduling', *In proc. of Fourth International Conference on Expert Systems in Production and Operations Management,* 134–144, (1990).
9. M.A. Salido, A. Giret, and F. Barber, 'Distributing Constraints by Sampling in Non-Binary CSPs', *In IJCAI Workshop on Distributing Constraint Reasoning,* 79–87, (2003).
10. E. Tsang, *Foundation of Constraint Satisfaction,* Academic Press, London and San Diego, 1993.
11. R. Wallace and E. Freuder, 'Ordering heuristics for arc consistency algorithms', *In Proc. of Ninth Canad. Conf. on A.I.,* 163–169, (1992).
12. T. Walsh, 'The constrainedness knife-edge', *In Proceedings of the 15th National Conference on AI (AAAI-98),* 406–411, (1998).
13. D.L. Waltz, 'Understanding line drawings of scenes with shadows', *The Psychology of Computer Vision,* 19–91, (1975).

To Block or Not to Block?

Alejandro González Romero and René Alquézar

Departament de Llenguatges i Sistemes Informàtics,
Universitat Politècnica de Catalunya,
C/Jordi Girona Salgado, 1-3, Mòdul C6,
08034 Barcelona- Spain
{aromero, alquezar}@lsi.upc.es

Abstract. The present work deals with the blocks world domain. While studying and analysing the theory about blocks world (BW) problems (BWP) an effective algorithm to solve instances was "discovered". This algorithm solves optimally many instances of blocks world problems and involves no search. The algorithm is backed up by ideas stated in Gupta and Nau (1991,1992) [5,6] and in Slaney and Thiebaux (1996,2001) [9,10]; the algorithm is suitable for programming and could serve to build an optimal block solver without forward search. The ideas behind this algorithm may constitute a basis for investigating other problems of more practical interest like the container loading problem and the bin packing problem.

1 Introduction

Blocks world planning has been widely investigated by planning researchers, primarily because it appears to capture several of the relevant difficulties posed to planning systems. It has been especially useful in investigations of goal and subgoal interactions in planning. The primary interactions studied have been deleted-condition interactions, such as creative destruction and Sussman's anomaly [2,3,4], in which a side-effect of establishing one goal or subgoal is to deny another goal or subgoal.

A problem in classical planning is given by a set of actions, an initial state and a goal. These problems can be formulated as problems of search from the initial state to the set of goal states by applying actions that map one state into another. The result of this search is a path in the state-space or a plan [7]. This problem is computationally hard and the best algorithms are bound to fail on certain classes of instances [1].

One response to overcome computational difficulties is to abandon search altogether and construct a special algorithm for a planning domain, incorporating in some way domain specific knowledge. This algorithm can be thought of as a mapping from any initial state and goal state to an action to be taken towards achieving the final goal; this approach has been called reactive planning. The algorithm presented in this work follows this line of thought.

2 Blocks World Theory

2.1 Preliminary Notions

A fixed finite set of *n* blocks is given. A *state* is an arrangement of these blocks on a table where each block is on another block or on the table; no more than one block is on a given block and no block is on more than one block. Given an initial state *I*, and a final (or goal) state *G*, the problem is to describe how to go from *I* to *G* preferably in the least number of steps. The permissible steps are:

1. Move onto the table a *clear block,* that is, one which has no block on it.
2. Move a clear block on top of another block.

A tower of height *n* of a state is a sequence $(b_1, b_2, ..., b_n)$ of blocks such that b_1 is clear, b_n is on the table and b_i is on $b_{i+1} (i=1,...,n-1)$. Notation: the state *s* whose towers are, for example, [1,5,4],[2] and [6,3] will be written *s*=[[1,5,4],[2],[6,3]]. Notice this is the same as, for example, [[6,3],[1,5,4],[2]] but not the same as [[5,1,4],[2],[6,3]].

Throughout this work we will be dealing many times with a certain group of predicates which describe a block's position. To be more precise lets look at, and analyse two examples.

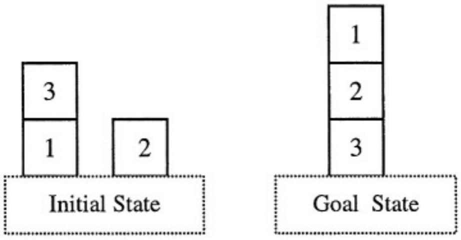


Fig. 2.1. Example of a block situation

In figure 2.1 an example of a situation can be seen (composed of an initial state and goal state) for the 3-blocks world. This situation could be described using a set of primitive predicates as follows:

$on\_table_s(1) \wedge on\_table_s(2) \wedge clear_s(2) \wedge on_s(3,1) \wedge clear_s(3)$
$on\_table_g(3) \wedge on_g(2,3) \wedge on_g(1,2) \wedge clear_g(1)$

Figure 2.2 illustrates an example of the support predicates $above_s(X,Y)$, $above_g(X,Y)$ and *well_placed(X).* The primitive or support predicates could also be negated as figure 2.2 shows in the example *not well_placed(1),* this will be usually written as ¬*well_placed(1).*

A *well_placed block* is a block whose initial position coincides with its goal position and all blocks below it are well_placed.

If a block *b* is on the table in the initial and goal states then *b* is a *well_placed block.*

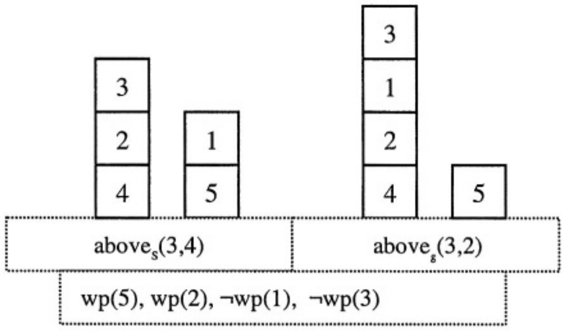


Fig. 2.2. Example of support predicates(wp= well_placed)

A *misplaced block* is a block that is not well_placed and a *misplaced tower* is a tower whose top block is misplaced. Therefore [3,2,4] in figure 2.2 is an example of a misplaced tower.

3 A Near Optimal Algorithm

3.1 The (I-II-III)-Algorithm

We will start by stating a therorem (Gupta and Nau, 1991) [5] which contains key features for our posterior analysis.

Theorem 3.1 (Gupta's and Nau's Theorem)
Let B=(I,G) be any solvable BW problem where Θ denotes the table , and P be any planfor B. Then there is a plan Q for B such that $|Q| <= |P|$ and Q has thefollowing properties:

1. *For every block b whose position in I is consistent with G, b is never moved in Q.*
2. *Q moves no block more than twice.*
3. *For every block b that is moved more than once, the first move is to the table.*
4. *For every block b that is moved to a location* $d \neq \Theta$, $on(b,d) \in G$.
5. *For every block b that is moved more than once, in the state immediately preceding the first move, no block whose position is inconsistent with G can be moved to a position consistent with G.*

With our terminology *1* would read:

1. A well_placed block *b* is never moved in a plan *Q*. Since the consistent notion is equivalent to the well_placed notion.

According to steps 1 and 2 of theorem 3.1, to go from an initial state *I* to a goal state *G* following an optimal path, well_placed blocks do not need to be moved and no block needs to be moved more than twice. This implies that, to calculate the distance from *I* to *G,* it suffices to find a set *S* of blocks of *I* which is precisely the set of blocks which are moved exactly once in an optimal path from *I* to *G*. Such a set *S* will be called a *1-set* (with respect to (*I,G*)) and an element of a 1-set a *1-block*. In general *S* is not unique, that is, there are examples of problems *(I,G)* such that there exist two different sets *S* and *S'* with the property that *S*, and also *S*', is a 1-set with respect to *(I,G)*. In figure 3.1 S={4} is a 1-set if 3 moves to the table and S'={3} is a 1-set if 4 moves to the table.

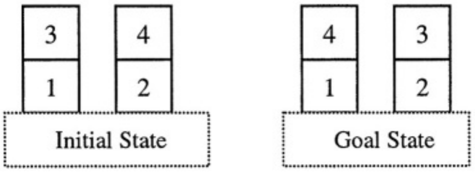


Fig. 3.1. Example of 1-sets and 2-sets

Alternately, to calculate the distance from *I* to *G*, it suffices to find a set *T* of blocks of *I* which is precisely the set of blocks which are moved exactly twice in an optimal path from *I* to *G*. Such a set *T* will be called a *2-set* and an element of a 2-set a *2-block*. Like *S*, *T* is not unique. In figure 3.1 *T*={3} is a 2-set if 3 moves to the table and *T'*={4} is a 2-set if 4 moves to the table.

It is easy to see that the distance from *I* to *G* is $m + |T|$ where *m* is the number of misplaced blocks and $|T|$ is the cardinality of a 2-set *T* with respect to (*I,G*).

Hence the problem of calculating the distance from *I* to *G* reduces to the problem of calculating $|T|$ (or $|S|$ since $|S| + |T| = m$).

Let us give now a useful definition:

Definition. A block *b* is locked (with respect to *(I,G))* if it is misplaced and there exists a block *b'* such that *b* is above *b'* in *I* and *b* is above *b'* in *G*. An example is shown in figure 3.2.

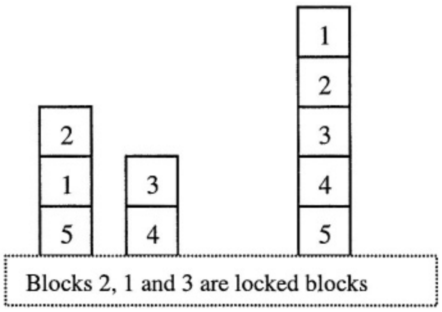


Fig. 3.2. Locked blocks examples

An important observation is that any locked block is moved twice in any path (=plan) from *I* to *G*; thus locked blocks are 2-blocks. In general 2-blocks are not locked blocks but in the special case in which *I* and *G* are towers, 2-blocks are the same as locked blocks and in fact the set of locked blocks is the unique 2-set.

It is now proposed an algorithm (the (I,II,III) – algorithm) which aims at finding an optimal path from *I* to *G*:

The (I-II-III)-Algorithm

I. If there is a constructive move (a move that increases the number of well_placed blocks), make such a move.
II. If there is no constructive move and there is a clear locked block, move such a block to the table.
III. If there are no constructive moves and no clear locked blocks, move a clear block with minimal deficiency to the table.

Let's now start the explanation about the step III of the (I,II,III)-algorithm (that is, put on the table a misplaced clear block with smallest *deficiency).* Essentially step III consists in unblocking a well placed or *proximal(=next-needed)* block by removing as few blocks as possible.

Definition. Let *S* be a state and *G* the goal state. A misplaced block *b* is *proximal*(with respect to *(S,G))* if there exists *x* such that *x* is a clear well_placed block or the table, and *b* is on *x* in *G*.

More informally, *b* is proximal(=next-needed) if, after removing the blocks above *b,* one can make a constructive move (on(b,x)). (A constructive move is one which increases the number of well_placed blocks). In a sense, proximal blocks are misplaced but close to being well_placed, if one removes the blocks above it.

The *deficiency* of a misplaced clear block *b def(b)* is defined as follows:

Definition. Let *S* be a state, *G* the goal state and b a misplaced clear block of *S* which is not proximal. If no blocks below *b* are well_placed or proximal define $def(b)=\infty$.

Otherwise $def(b) = n-r$ where *n* and *r* are as follows:

1. $b_1,\dots,b_n$ are *n* blocks such that $b_1 = b$ and b_j is on $b_{(j+1)}$ in S $(j=1,\dots,n-1)$.
2. $b_1,\dots,b_n$ are neither well_placed nor proximal.
3. b_n is on a well_placed or proximal block in *S*.
4. *r* is the number of blocks in $b_1,\dots,b_n$ which are locked.

In other words, the deficiency of *b,* in case it is not ∞, is the number of blocks that are needed to remove from the tower to which *b* belongs, to reach a well_placed or proximal block, with the proviso that locked blocks do not count.

Step III now reads: If no constructive moves can be made and no clear block is locked then put a misplaced clear block of smallest deficiency on the table.

Notice that the (I-II-III)-algorithm is complete since a misplaced clear block will always have a deficiency associated with it.

The following is equivalent to theorem 3.1 (Gupta and Nau).

There is an optimal path(=plan) from I to G in which the following rules are respected:

1) Never move a well_placed block.
2) If a constructive move (a move that increases the number of well_placed blocks) can be made, make it; if not move a misplaced block to the table. (Notice that 2 implies 1).

One can see that in a plan respecting 1) and 2) no block is moved more than twice.

This theorem justifies step I in the (I-II-III)-algorithm. If in a state more than one constructive move can be made it does not matter which one is chosen. However, the main problem is:

In order to build a near optimal plan that is optimal for many blocks world problems, if no constructive move can be made, **which** block must be moved to the table?. This is answered by steps II and III of the (I,II,III) algorithm.

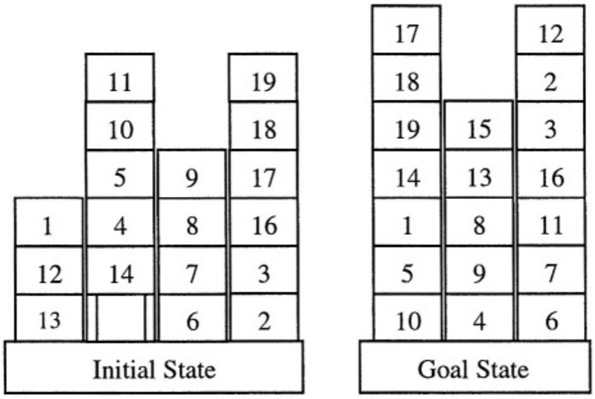


Fig. 3.3. A 17-block example

When applying algorithm I-II-III to the example shown in figure 3.3, the following analysis is conceived:

Since there are no constructive moves step I can not be used; step II can not be applied either since there are no locked blocks. Therefore step III is our last hope, and it solves the problem since the deficiencies of the blocks 1,11,9 and 19 are ∞, 1, 2, and ∞ respectively, therefore step III suggests to move 11 to the table since it is the block with lowest deficiency (1). The rest of the moves are constructive moves (step I) and the number of moves it takes to go from the Initial State to the Goal State is 18 which is optimal. In fact the optimal path is unique.

3.2 The (I-II)-Algorithm

This section shows that problems where the initial or goal state has less than two misplaced towers of height > 1 are solved by the (I-II)-Algorithm (defined below) and are therefore easy. Also the distance from I to G in such problems is the sum of the num-her of misplaced blocks and the number of locked blocks. Notice that problems where the initial or goal state consists of a single tower have this property.

Definition. A problem *(I,G)* will be called *critical* if no constructive moves can be made and no clear block in *I* is locked.

Proposition 1. If *(I,G)* is critical then both *I* and *G* have at least two misplaced towers of height > 1.

Proof. Since no constructive moves can be made, by Slaney and Thiebaux (1996) page 1210 [9], there is a *clear deadlock* $(m_0, m_1,\ldots,m_n)$ in *(I,G)*. This means that the blocks m_i are clear in *I* and misplaced, $m_0=m_n$ and $Below\_G(m_i)$ intersects $Below\_I(m_{i+1})$ $(i=0,1,\ldots,n-1)$ where *Below_S(b)* is the set of blocks below *b* in *S*. This implies that no m_i is on the table in *S* and no m_i is on the table in *G*.

We claim that $m_i \neq m_{i+1}$. If we had $m_i = m_{i+1}$ then $Below\_G(m_i)$ would intersect $Below\_I(m_i)$, which is impossible because m_i is not locked. Hence $m_i \neq m_{i+1}$.

Therefore the tower whose top is m_i and the tower whose top is m_{i+1} are two different misplaced towers of height > 1. Thus *I* has the desired property.

Now, suppose *G* has less than two misplaced towers of height > 1. Then all m_i belong to a single tower *T* of *G*. We claim that m_i is above m_{i+1} in $G\ (i=0,1,\ldots,n-1)$. Let *b* be a block below m_i in *G* and below m_{i+1} in *I*. Clearly *b* is in *T*. The block m_{i+1} cannot be above b_i in *G* since m_{i+1} is not locked. Hence m_{i+1} is below *b* and, therefore, below m_i in *G*. Thus, in *G*, m_1 is above m_n. But this is impossible because $m_1=m_n$. Therefore *G* has at least two misplaced towers of height > 1. □

The (I-II)-Algorithm

I. If there is a constructive move (a move that increases the number of well_placed blocks), make such a move.
II. If there is no constructive move and there is a clear locked block, move such a block to the table.

This algorithm is not complete in general, of course, because it gets stuck when one arrives at a state *I* such that (*I,G*) is critical. Proposition 1 states that this never happens if the initial or goal state has less than two misplaced towers of height > 1 (in particular this holds if *I* or *G* consists of a single tower).

Proposition 2. Let *(I,G)* be a problem such that *I* or *G* has less than two misplaced towers of height > 1. Then the (I-II)-algorithm finds an optimal path from *I* to *G* and the distance from *I* to *G* is: (number of misplaced blocks)+(number of locked blocks).

By proposition 1, *(I,G)* is not critical and therefore the (I-II)-algorithm can be applied initially. But notice that if this algorithm is applied to a state with less than two misplaced towers of height > 1, the resulting state has the same property. It follows that the algorithm can be applied at any stage. Since the moves of the (I-II)-algorithm are good, that is, they get us closer to *G*, we eventually reach the goal following an optimal path. The number of moves of this path is the number of constructive moves + the number of locked blocks. This is the distance from *I* to *G*.

3.2 The 5-Blocks World

Most of the 5-block world problems are solved using the (I-II)-algorithm since their initial or goal state has less than two misplaced towers of height > 1. It is then desired to find problems with 5 blocks in which no step I (=constructive) or step II (=locked) move is possible.

It can be assumed, by renumbering the blocks if necessary, that any problem with 5 blocks is equivalent to one in which the goal state is one of the following seven:

1. [1,2,3,4,5]
2. [[1,2,3,4], [5]]
3. [[1,2,3], [4], [5]]
4. [[1,2], [3], [4], [5]]
5. [[1], [2], [3], [4], [5]]
6. [[1,2], [3,4], [5]]
7. [[1,2,3], [4,5]]

If the goal is one of the first five, there is always a step I or a step II move available.

If the goal state is [[1,2], [3,4], [5]] and the initial state is one of:

1. [[3,2], [1,5,4]]
2. [[3,2], [1,4,5]]
3. [[3,2], [1,4], [5]]

then no step I or step II move is possible.

If the goal state is [[1,2,3], [4,5]] and the initial state is one of:

4. [[4,3,2], [1,5]]
5. [[4,3,1], [2,5]]
6. [[2,5,1], [4,3]]
7. [[2,1,5], [4,3]]
8. [[2,5], [4,3], [1]]
9. [[1,5], [4,3], [2]]
10. [[1,5], [4,2], [3]]

then no step I or step II move is possible.

These are the only situations with 5 blocks in which no step I or step II move is possible. In any of them any step III move is optimal. Thus the (I-II-III)-algorithm is 100% effective in solving optimally the 5-blocks problem.

4 Conclusions and Future Work

The (I-II-III)-algorithm can solve all 5 block world problems (bwp) in an optimal fashion. It can also solve optimally many other classes of bwp as can be seen from the example previously shown consisting of 17 blocks. To solve examples where the initial state, or the goal state, have less than two towers of height greater than one, it is sufficient to employ the (I-II)-algorithm.

Like it was done for Slaney's and Thiebaux's algorithm (GN2 with Δ sequence) in [10] the (I-II-III) algorithm can be programmed to build a near optimal Blocks world solver. Once built this algorithm could be tested using BWSTATES, a program built by Slaney and Thiebaux which generates random BW states with uniform distribution (see [10] pages 122-128).

Besides being an optimal algorithm for a large number of instances, it enlightens the crucial key ideas that should be learned for this domain. In particular it helps to know that certain rules, when fired, not only approach the goal, but approach optimally to it, for instance putting clear locked blocks on the table.

BW's place as an experimentation benchmark constitutes a basis for investigating other problems of more practical interest. The abstract problem of which BW is an instance is the following: sets of actions producing the goal conditions (constructive actions) cannot be consistently ordered so as to meet all of their preconditions (that is, step II and step III actions are needed). To resolve each locked block situations and minimal deficiency block situations, a number of additional actions have to be introduced (non-constructive actions (step II and step III)). Since these situations are not independent, there is a need to reason about how to resolve them all using as few additional actions as possible. This core problem is present in other more realistic situations. For instance, moving blocks is not very different from moving more exciting objects such as packages, trucks and planes. Again, the version of BW in which the table has a limited capacity captures the essence of the container loading problem, a problem that is crucial to efficiency of freight operations [8,11]. Therefore, finding the right generalisations of strategies that are effective for BW appears promising as an approach to more sophisticated problems.

Acknowledgement

The first author acknowledges the funding of CONACYT.

References

1. Bylander, T. , The computational complexity of strips planning. *Artificial Intelligence,* 69 (1994): 165-204.
2. Charniak, E. And McDermott, D. , Introduction to Artificial Intelligence. *Addison-Wesley,* Reading, MA.,1985.
3. K. Erol, D. N. and Subrahmanian, V. , When is planning decidable?, in *Proc. First International Conference AI Planning Systems,* 1992.
4. R.L. Graham, D.K. and Patashnik, O. Concrete Mathematics: a Foundation for Computer Science. *Addison-Wesley,* 1989.
5. N. Gupta, D.S. Nau, Complexity results for Blocks World planning, in: *Proc. AAAI-91,* Anaheim, CA, 1991, pp. 629-633.
6. N. Gupta, D.S. Nau, On the complexity of Blocks World planning, *Artificial Intelligence 56* (1992) 223-254.
7. Nilson, N.J. Principles of Artificial Intelligence. Tioga, Palo Alto 1980.

8. A.E. Rizzoli, L.M. Gambardella, M. Zaffalon, M. Mastrolilli, Simulation for the evaluation of optimised operations policies in a container terminal, in: *Proc. HMS99, Maritime and Industrial Logistics Modelling and Simulation,* Genoa, Italy, 1999.
9. J. Slaney, S. Thiébaux, Linear time near-optimal planning in the Blocks World, in: *Proc. AAAI-96,* Portland, OR, 1996, pp. 1208-1214.
10. J. Slaney and S. Thiébaux, Blocks World revisited, *Artificial Intelligence 125* (2001): 119-153.
11. T. Slavin, Virtual port of call, in: *New Scientist,* June 1996, pp. 40-43.

Adaptive Penalty Weights When Solving Congress Timetabling

Daniel Ángel Huerta-Amante and Hugo Terashima-Marín

Center for Intelligent Systems, ITESM, Campus Monterrey
Ave. Eugenio Garza Sada 2501 Sur, C. P. 64849,
Monterrey, Nuevo León, Mexico
Tel. +52 (81) 8328 4379
dhuerta@csi.mty.itesm.mx, terashima@itesm.mx

Abstract. When a Genetic Algorithm is used to tackle a constrained problem, it is necessary to set a penalty weight for each constraint type, so that, if the individual violates a given constraint it will be penalized accordingly. Traditionally, penalty weights remain static throughout the generations. This paper presents an approach to allow the adaptation of weights, where the penalty function takes feedback from the search process. Although, the idea is not new since other related approaches have been reported in the literature, the work presented here considers problems which contain several kinds of constraints. The method is successfully tested for the congress timetabling problem, a difficult problem and with many practical applications. Further analysis is presented to support the efficiency of the technique.

1 Introduction

Genetic Algorithms (GAs) [1, 2] are an optimization technique that has shown very good performance for a variety of combinatorial and constrained optimization problems [3]. Individuals in a population are evaluated by a fitness function, which along with other operators such as selection, crossover and mutation, guide the search to promising regions of the solution space. For a constrained problem, the fitness function usually is represented as a penalty function in which the penalty weights for each type of constraint are set at the beginning and remain static throughout the generations. The penalty function computes the degree of constraint violation for a solution, and this is used to penalize each individual accordingly.

The main drawback of this approach, however, is precisely the arbitrary selection of these coefficients. Usually, these weights are set in regards to the importance of the constraints. A different alternative is to run a series of experiments in order to tune the weights. But, what is the problem when proposing these weights at the beginning? It is not necessarily true that setting a high penalty for a constraint would mean that the final solution will satisfy that constraint; sometimes it happens that giving a low penalty to an important constraint will result in a solution in favor of that constraint. Would it be possible to forsee all these cases? There is an alternate path for tackling these problems by using a set of techniques to adapting weights incorporating feedback from the search process. A complete survey of these strategies are presented and summarized by Michalewicz and Schmidt [4]. The article addresses issues about the general nonlinear programming problem and the emerging approaches for solving it. The idea is to adjust the penalties according to the difficulty of the given constraint in a given stage of the solution process. Usually, constraints can be of two kinds: equalities and inequalities. In general terms, inequalities are easier to solve. For instance, if a variable x requires to be assigned a value less or equal to zero, then its domain is $[-\infty, 0]$. In the contrary, for an equality a specific value has to be assigned to a given variable.

Let us assume we have constraint R_1. When a solution Z satisfies that constraint, then Z is feasible with respect to the constraint. Otherwise Z is considered infeasible. In evolutionary computation, the fitness function serves as a way for rating individuals (solutions) in the population. Some individuals may be feasible whereas others are infeasible. A percentage of feasible solutions can be computed for a particular population and this can be used (1) to detect the hardness of a given constraint, that is, the less feasibility the more difficulty of that constraint; and (2) to keep a percentage of feasible individuals for a given constraint, adjusting that percentage by applying genetic operators. It is known that usually an optimal solution for a problem is surrounded by a number of infeasible solutions, or sometimes the optimal solution can be found around the boundary between feasible and infeasible regions [5]. Hamida [6] has proposed a technique which is based on the feasibility percentage. This idea was extended in this paper to solve a more constrained problem such as the congress timetabling.

The goal of the work presented here is to introduce adaptability of penalty weights to a highly constrained problem with different kind of constraints, and by doing so, to provide a general procedure for including adaptability to a contrained problem.

This article is organized as follows. Section 2 presents the methodology followed to carry out this investigation. Section 3 establishes the experimental set up, the results and their discussion. Finally, in section 4 we include our conclusions.

2 Methodology

In what follows we describe the key steps followed to achieve the goal of this investigation. First, the Congress Timetabling Problem is defined and the implementation details of a problem generator are presented. Next, the adapted Genetic Algorithm, its representation and parameters are introduced. Based on the original algorithm for adapting penalty weights, two new algorithms were designed taking into account particular features of the congress timetabling problem and its constraints.

2.1 The Congress Timetabling Problem

The problem of assigning events or activities to resources such as time slots, physical space or personnel, in such a way that various constraints are satisfied, is known as the timetabling problem [7]. More formally, the problem can be described as a set A of events, a set D of time slots, and a set C of constraints. The aim is to assign the events into the time slots satisfying all the constraints [8]. For a more complete survey on timetabling refer to the work by Schaerf [9]. For this research, a specific kind of timetabling problem was chosen, the congress timetabling problem with the following features:

1. In most of the timetabling problems it is assumed that all time slots are of the same duration. In our problem, duration of time slots is variable.
2. There is no event overlapping, i.e. there exists only one event for every unit of time.
3. The domain of an event consists in a set of minutes, instead of a time lapse.
4. The constraints that are considered in this work are PRESET, EXCLUDE, ORDER and TIME. PRESET(x,y) establishes that event x should be scheduled at exactly y hours. EXCLUDE(x,$*a$) in which event x must not happen at certain times given in array a. ORDER(x,w) which requires an event x to be scheduled *just* before some other event w. TIME(t) indicates that the sum of all scheduled events should not be greater than t.

2.2 Generator of Congress Timetabling Problems

A problem generator was implemented to create an experimental testbed. There are several parameters that were established according to features found in real congress timetabling problems. The number of days for a congress can be between 3 and 5 days; duration of each day varies between 450 and 675 minutes, depending on the number of activities per day that can be between 15 and 35, and the activity duration that can be between 1 and 90 minutes. 80% of the activities are involved in at least one constraint. If an event is constrained by PRESET, then that event can not involve other kind of constraint. There is a probablity of 0.25 for getting a PRESET constraints, 0.45 for EXCLUDE, and 0.30 for ORDER.

2.3 Solving the Problem with GAs

Given that activities must not overlap and the goal is to find a good ordering of events, it is natural to use a permutation-based representation. This kind of representation has been studied extensively in the literature [1, 10]. Each chromosome represents a complete solution to the problem with n activities to be scheduled in the given order. Each event is assigned one by one until a day is complete. Three variations were tested for doing this:

CGA1.- It sequentially assigns activities until there is one that exceeds the time limit of the day being filled. The next activity is then scheduled in the following day.

CGA2.- Similar to CGA1, but the exceeding activity is scheduled in the following day.

CGA3.- In addition to CGA2, the PRESET constraint was considered. If an activity involves a constraint of this kind, then it is is scheduled at the time requested, if possible. Otherwise, it is shifted to the right to the first interval in which it fits.

The objective function used to evaluate each chromosome x is:

$$f(x) = \frac{1}{1 + P(x)} \tag{1}$$

where

$$P(x) = c_{time}p_{time} + c_{preset}p_{preset} + c_{order}p_{order} + c_{exclude}p_{exclude} \tag{2}$$

where p_{time}, p_{preset}, p_{order} and $p_{exclude}$ are the penalty weights for each constraint type, and c_{time}, c_{preset}, c_{order} and $c_{exclude}$ are the degrees of violation for each constraint.

It is expected that the chromosome provides an order resulting in a schedule with time equal to the total duration of the congress. This is the constraint TIME. If not, the individual is penalized in relation to the extra minutes used. The PRESET constraint is penalized with the difference in minutes between the actual scheduled time and the time in which the event should have been scheduled. The ORDER constraint is penalized by the number of activities necessary to skip in order to have the first activitiy before the second one. To penalize the constraint EXCLUDE, the number of minutes overlapping the prohibited time lapse was used.

Penalty weights for the constraints were set with respect to the their importance following the advise of a human expert in congress scheduling along with experimental tuning. The weight for the TIME constraint is 0.01; 0.005, 0.003 and 0.05 for PRESET, EXCLUDE, and ORDER constraints, respectively.

2.4 Parameter Tuning

It is known that a GA usually needs a process of tuning to identify the parameter set that behaves best against a set of problems. To do this, 9 random problems were generated, and each was tested with 15 runs for different parameter configurations (population size, selection, crossover, mutation, and number of generations). Table 1 shows the best configuration for each GA with static penalty weights. Variation CGA2 performed better than the other two. This algorithm was selected to include the adapting model for penalty weights. In the following sections CGA2 will be renamed as CGA.

2.5 Algorithm for Adaptive Penalties

The original method suggested by Hamida and Schoenauer [6] for adapting penalty weights modifies the penalty coefficients according to the proportion of feasible individuals in the population. Their strategy is used to combine feasible with infeasible individuals in order to explore the region around the feasible domain, and it takes into account multiplication and division operations for adjusting the penalties. The static parameter *fact* is the constant that is used to multiply (to increase) or divide (to decrease) the current penalty to compute the

Table 1. Final parameter set for each GA with static penalty weights

parameter/algorithm	CGA1	CGA2	CGA3
Selection	Tn	Tn	Tm
Tournament size	3	3	4
Crossover	GOX	GPX	GPX
Crossover probability	0.95	1.00	0.90
Mutation	Swamp	Order	Order
Mutation probability	0.65	0.34	0.34
Population size	350	350	200
Fitness average	0.545264	0.650261	0.201606

new penalty that will be used in the next generation (or in a set of generations). *tTarget* is another parameter to determine the target portion of feasible individuals in the population. The original technique adapted for our problem with several kinds of constraints was called CGAA1.

The effect of the parameter *fact* is the same for adjusting the penalty, regardless how close the the proportion of feasible individuals is to *tTarget.* In other words, there is no difference in the adjustment if a constraint is about to meet the target portion of feasible individuals, or when it is far from it. In order to correct this problem the variable *error* was introduced, which is the difference between the actual portion of feasible individuals *(tT)* and the target portion *(tTarget): error = tT – tTarget.* So, the two suggested rules by Hamida and Schoenauer are merged into one:

$$\alpha_i[t+1] = \alpha_i[t] - (tTrain \times error) \qquad (3)$$

$\alpha_i[t]$ is the parameter that is adjusted depending on the *error* for a constraint of type i. If *error* is positive (there are more feasible individuals than the target), $\alpha_i[t]$ is decreased. Otherwise, $\alpha_i[t]$ is increased. $\alpha_i[0]$ starts with a value equal to the penalty weight (used in the static algorithm) for constraint i. The constant *tTrain* helps to adjust the penalty in terms of the penalty units, which in this case are in the order of 10^{-3}. This technique was called *penalty adaptability based on error* or CGAA2.

2.6 Adaptation Parameters

In order to improve the adapting process for penalty weights some additional parameters were used, for both the CGAA1 and CGAA2, having in mind the different types of constraints involved in the congress timetabling problem.

Relaxation. The aim is to transform an equality into an inequality and transform this gradually back into an equality. For instance, $R_2 \longrightarrow x = 5$ represents a constraint called R_2 that is limiting the domain of variable x to $\{5\}$; using relaxation, the domain of x is [(5 – *relaxation*), (5 + *relaxation*)] as shown by equation $R_2 \longrightarrow (5 - relaxation) < x < (5 + relaxation)$. To determine the initial relaxation, it was necessary to analyze the equality constraints: TIME and PRESET. With respect to TIME, it was observed that some particular features of the problem are related to the initial behavior of TIME. A backpropagation neural network of 4 inputs and 1 output was used to establish the initial relaxation. The inputs are the total duration of congress, the number of days in the congress, the number of activities in the congress, and the portion of feasible individuals in the initial population; the output is the necessary relaxation that produces, for a given problem, the right portion of feasibles in the initial population for the TIME constraint. For the constraint PRESET a particular relationship between the problem characteristics and its behavior was not found, but the average duration of the activities was used for relaxing this constraint. To make sure that the given relaxion was right for this problem, this was still slightly adjusted. Various multiplications by constants were used. For TIME the constants were: 0.5, 1.0, 1.5 y 2.0. For PRESET: 0.5, 1.0, 2.0, 3.0 y 4.0.

tMax. When the proportion of individuals is greater than *tMax* the relaxation value is reduced by 1.

tRel. It is reasonable to think that when a solution is feasible with respect to a specific constraint thanks to relaxation, the individual should not be penalized, or at least not as hard as an infeasible solution. This parameter was introduced to test different alternatives for penalizing this kind of individuals: (1) penalize all individuals according to their actual degree of violation, (2) do not penalize these individuals, but penalize the infeasible in relation to the actual degree of violation, (3) do not penalize these individuals, but penalize the infeasible in relation to the difference between the actual degree of violation and the relaxation, and (4) penalize these individuals with a function *(relaxation/degree)* and penalize the infeasibles with (*degree – relaxation* + 1).

Lapse. The idea behind this parameter comes from the work reported by Eiben for graph coloring [11]. In this model, called *Stepwise Adaptation of Weights,* the parameter is used to indicate the number of generations (or evaluations since there is only one individual in the population) before coloring the node with the highest penalty, and update the penalties for the rest of the nodes. Whereas this model focuses in the nodes, in our work we concentrate this effort in the constraints. When the adaptations are large, it is required to give some time (certain number of generations) to the GA to react to the adjustments. *Lapse* represents this number of generations and has a value between 1 and 25.

tEnd. It specifies the termination criterion for adaptation. Search in a GA with adaptive parameters behaves well for a number of generations; however, often it reaches a point where the average fitness starts to decrease. It is important to have a parameter to stop adapting. *adapting zone* is a region (set of generations) in which the constraints behave unsteady. After passing this zone the constraints suffer small changes, except the constraint PRESET, and this is used in their advantage to find better individuals. Our conjecture is that at the end of the *adapting zone* we should stop adapting. To detect the end of the *adapting zone* the following steps are suggested: (1) for each generation and for each constraint, find the variance on the behavior of the best found so far in the last 10 generations, (2) compute the average variance for each constraint, and (3) if this average is less than *tEnd* then the *adapting zone* has ended. *tEnd* has a value between 0.000500 y 0.000005. Behavior of this parameter for a particular instance is shown in Figure 1, where the grey region is the *adapting zone.* After experimentation, it was determined that this parameter was only helpful for algorithm CGAA1.

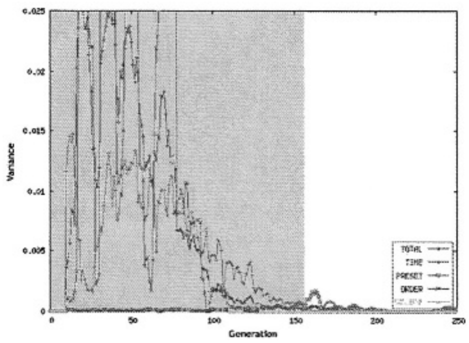


Fig. 1. Variance on the behavior of each constraint type during search

3 Experimental Results and Discussion

To introduce adaptation to the constraints in our problem, the following steps were used: (1) tune the parameters separately for the equality constraints (TIME and PRESET), (2) introduce adaptability to each of the equality constraints, (3) introduce adaptability to the rest of the constraints, from the hardest to the easiest, and finally, (4) tune the *tEnd* parameter. The parameter order for tuning is as follows: *Lapse, relaxation, fact/tTrain, tMax* and *tRel* (only for TIME and PRESET), and *tTarget* (just for EXCLUDE and ORDER). Three problems were designed (generated) for each duration of 3, 4 and 5 days. 15 runs were carried out for each problem. Tables 2 and 3 show the final tuning for the set of parameters.

Results obtained with these combinations are presented in Table 4 for all three algorithms tested. Each algorithm reports results for problems with 3, 4 and 5 days in the duration of the congress. 17 different problems were used for each duration, with 30 runs for each instance. It is observed that for constraints TIME, PRESET, ORDER and EXCLUDE, in both algorithms CGAA1 and CGAA2, the adaptive process produces a decrease in the constraint violation. For instance, for constraint TIME the average number of minutes is 10.00 for CGA (static penalty weights) in congresses lasting 4 days, while for CGAA1 decrease to 9.67, and, for CGAA2 there is still a slight improvement decreasing to 6.06. In the last column of the table FITNESS is reported. In order to have

Table 2. Parameters for algorithm CGAA1

parameter/constraint	TIME	PRESET	ORDER	EXCLUDE
tMax	0.70	-	-	-
lapse	8	-	8	8
relaxation	0.50	-	-	-
fact	1.1	-	1.1	1.1
tRel	1	-	-	-
tEnd	0.0002	-	0.0002	0.0002
tTarget	0.50	-	1.00	1.00

Table 3. Parameters for algorithm CGAA2

parameter/constraint	TIME	PRESET	ORDER	EXCLUDE
tMax	0.70	0.50	-	-
lapse	1	20	1	1
relaxation	0.50	1.00	-	-
tTrain	0.05	0.10	0.05	0.05
tRel	1	1	-	-
tEnd	-	-	-	-
tTarget	0.50	0.50	1.00	0.98

Table 4. Table comparing algorithms CGA, CGAA1, and CGAA2 for different durations of the congress

	Days	TIME	PRESET	ORDER	EXCLUDE	FITNESS
CGA	3	3.43	68.83	0.03	7.33	0.726058
	4	10.00	102.70	0.03	5.80	0.654762
	5	17.10	91.60	0.40	6.40	0.618476
CGAA1	3	2.47	67.60	0.00	2.73	0.736826
	4	9.67	88.87	0.00	5.16	0.680868
	5	12.36	86.40	0.26	5.73	0.647106
CGAA2	3	1.63	67.60	0.00	4.60	0.760208
	4	6.06	88.86	0.00	5.63	0.671713
	5	12.93	91.66	0.00	3.10	0.639005

a point for comparison against the CGA, the best individual in the adaptation algorithms for a particular run was evaluated using the static weights.

Table 5 shows results for a GA with static penalties, compared against the adpating models such as the CGAA1 and CGAA2 for 50 randomly generated problems (30 runs for each). It is observed that the algorithm CGAA1 behaves better in general than the static model, and algorithm CGAA2 behaves slightly better than both. The degree of violation is shown for each kind of constraint (minimizing) and in the right column the average fitness (maximizing).

Table 5. Table comparing performance of algorithms CGA, CGAA1, and CGAA2 for 50 randomly generated problems

Algorithm	TIME	PRESET	ORDER	EXCLUDE	FITNESS
CGA	10.29	95.75	0.56	6.21	0.64570
CGAA1	8.47	93.65	0.37	1.92	0.66239
CGAA2	7.85	98.16	0.19	1.20	0.66287

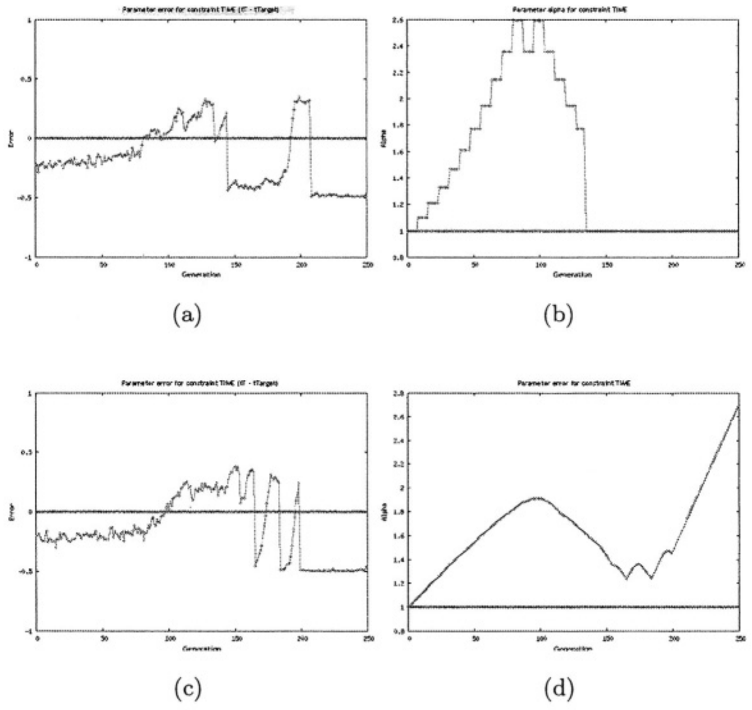


Fig. 2. Behavior of *error* and a for TIME in CGAA1(a,b) and CGAA(c,d)

Figure 2 present the behavoir of the *error* along with parameter α for constraint TIME for both algorithms CGAA1 and CGAA2, respectively. It is observed that when *(tT – tTarget)* is negative, the algorithm increases the penalty so that the number of feasibles also increases, and when the actual number of feasibles is greater than *tTarget* is greater than 0, it tries to pull it back. IN CGAA1, when α equals 1 around generation 150, it means that adaptation has been stopped. Adaptation is not halted in CGAA2.

4 Conclusions and Future Work

It has been demonstrated through an empirical study that adapting penalty weights in a GA for a constrained problem with multiple types of constraints performs better than a GA with static penalty weights. Despite this fact, adapting weights requires to set and tune various parameters. Future work is suggested to investigate more about this trade-off. Several lessons were learned with this experimentation that could help to tackle similar problems in the future. For instance, the parameter *tRel* should not be tuned. For parameter *tTarget,* it was found that for equality constraints should be 0.50, and for inequality this value is around 0.90. Parameter *tMax* varies from around *tTarget* to *tTarget*+0.20. This work also proposed a set of steps to introduce adaptability; it would be interesting to prove if they would help when facing other similar problems.

Acknowledgments

This research was supported in part by ITESM under the Research Chair CAT-010 and the Conacyt Project under grant 41515.

References

1. D. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley Publishing Company, Inc., 1989.
2. Melanie Mitchell. *An Introduction to Genetic Algorithms: Complex Adaptive Systems.* MIT Press, 1998.
3. R. Sarker, M. Mohammadian, and Y. Xin. *Evolutionary Optimization.* Kluwer International Series. 2002.
4. Zbigniew Michalewicz and Martin Schmidt. Evolutionary algorithms and constrained optimization. *Evolutionary Optimization,* pages 57–86, 2002. Sarker, R. and Mohammadian, M. and Xin, Y., editors.
5. Hatem Hamda and Marc Schoenauer. Adaptive techniques for evolutionary topological optimum design. In *ACDM'2000,* 2000.
6. S. Ben Hamida and Marc Schoenauer. An Adaptive Algorithm for Constrained Optimization Problems. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J.J. Merelo, and H.-P. Schwefel, editors, *Proceedings of 6th PPSN,* pages 529–538, Heidelberg, Germany, September 2000. Paris, Springer-Verlag. LNCS Vol. 1917.
7. J. Lee, Y. Fanjiang, and L. Lai. A software engineering approach to university timetabling. *International Symposium on Multimedia Software Engineering,* pages 124 – 131, 2000.
8. H. Terashima. *Combinations of GAS and GSP strategies for solving examination timetabling problems.* PhD thesis, Instituto Tecnologico y de Estudios Superiores de Monterrey, October 1998.
9. A. Schaerf. A Survey of Automated Timetabling. *Artificial Intelligence Review,* 13:87–127, 1999.
10. Lawrence Davis. *Handbook of Genetic Algorithms.* Thompson Computer Press, 1996.
11. A. E. Eiben, J. K. van der Hauw, and J. I. van Hemert. Graph Coloring with Adaptive Evolutionary Algorithms. *Journal of Heuristics,* 4(1):25–46, 1998.

Decomposition Approaches for a Capacitated Hub Problem

Inmaculada Rodríguez-Martín and Juan-José Salazar-González

DEIOC, Universidad de La Laguna,
38271 La Laguna, Tenerife, Spain
{irguez, jjsalaza}@ull.es

Abstract. In this work we address a capacitated hub problem arising from a Telecommunications application. In this problem we must choose the routes and the hubs to use in order to send a set of commodities from sources to destinations in a given capacitated network with a minimum cost. The capacities and costs of the arcs and hubs are given, and the graph connecting the hubs is not assumed to be complete. We present a mixed integer linear programming formulation and describe three different decomposition techniques to get better performances than simply using a direct general solver on the model. These approaches can be applied to deal with more general problems in Network Design.

1 Introduction

This work concerns the following optimization problem arising in the design of a Telecommunications system. Let us consider a set I of computers (*source terminals*) that must send information to a set J of computers *(destination terminals)* through a network. The network consists of cables, possibly going through some computers *(routers)* of another set H. The whole set of computers is called *node set* and it is represented by $V := I \cup J \cup H$, and the set of cables is called *arc set* and it is represented by A. Hence, we have a directed graph $G = (V, A)$. A given computer can be at the same time source terminal, destination terminal and router, thus subsets I, J and H are not necessarily disjoints. Based on the application motivating this work, we will assume in this paper that the three computer sets are disjoint. When a cable allows communication in both direction it is considered as two opposite arcs in A. We will assume there are not arcs in A from a destination terminal to a source one, i.e., there are no arcs from nodes in J to nodes in I.

Associated to each arc $a \in A$ there is a *capacity* q_a, representing the maximum amount of information units that can go through it, and a value c_a representing the cost of sending a unit of information through a and called *routing cost.* Equally, associated to each router $h \in H$ there is a *capacity* q_h, and a value c_h representing the fix cost of using router h and called *maintenance cost.* Finally, for each pair $(i,j) \in I \times J$ of terminals we are given an amount d_{ij} of information to be sent from i to j. Since most of these values can be zero, the d_{ij} information units going from i to j will be termed *commodity* if $d_{ij} > 0$, and for simplicity they will be referred with one index $k \in K := \{(i,j) \in I \times J : d_{ij} > 0\}$. Also for brevity of notation, we will write d_k instead of d_{ij} if $k = (i,j)$.

The optimization problem, called *Capacitated Hub Problem* (CHP), consists in deciding the way the required flow of information between terminals must traverse the capacitated network in order to minimize the sum of the routing and maintenance costs.

A particular case of this problem, referred as *Capacitated Multiple Allocation Hub Location Problem,* has been recently considered in Ebery, Krishnamoorthy, Ernst and Boland [6], motivated by a postal delivery application. In this particular combinatorial problem, where the name "router" is replaced by "hub" and "terminal" by "client", the capacity requirements only concern the hubs and apply only to mail letters coming directly from a client and going directly to a client. Moreover, the maintenance cost of a hub is paid when it receives or deliveries letters to a client, and not paid if the letter is moved from one hub to another hub. Those assumptions allow to work with a complete graph between hubs and to reduce the potential paths for a letter (as a letter will never go through more than two hubs). Paper [6] presents mathematical models exploiting this advantages by using variables based on 3 and 4 indices. Another problem closely related to CHP has been addressed in Holmberg and Yuan [10]. Uncapacitated versions of the CHP have been extensively studied in the literature (see, e.g., Campbell [4], Ernst and Krishnamoorthy [7], Mayer and Wagner [12], Skorin-Kapov, Skorin-Kapov and O'Kelly [15]). The CHP is also related to network design (see, e.g., Barahona [2], O'Kelly, Bryan, Skorin-Kapov and Skorin-Kapov [14]).

In our application from Telecommunications (see Almiñana, Escudero, Monge and Sánchez-Soriano [1]) the above mentioned assumptions do not hold since we have maintenance costs and capacity constraints also for routers not connected with terminal nodes. We do not know any previous study on the CHP as it is here defined, and we present a mathematical formulation based on 2-index variables. We use this model to develop an algorithm for CHP. Some preliminary computational results are also presented.

2 ILP Model

To simplify the notation, it is convenient to extend the arc set of G with the dummy arcs from destinations to origins, i.e.,

$$A := A \cup \overleftarrow{K},$$

where $\overleftarrow{K}$ is the set of arcs from $j \in J$ to $i \in I$ if $d_{ij} > 0$. For each subset $S \subset V$, we will denote by

$$\delta^+(S) := \{(u,v) \in A : u \in S, v \in V \setminus S\}$$

and

$$\delta^-(S) := \{(u,v) \in A : u \in V \setminus S, v \in S\}.$$

Let us consider a decision variable y_h associated to each $h \in H$ that takes value 1 when the router is used, and a continuous variable x_a associated to each $a \in A$ representing the amount of communication traversing cable a.

To present a mathematical model we also make use of an additional set of continuous variables, $[f_a^k : a \in A]$, for each commodity $k \in K$. If $k = (i,j)$ is the commodity going from i to j, then variable f_a^k represents the amount of communication from source i to destination j traversing cable a. Then the mathematical model is:

$$\min \sum_{a \in A} c_a x_a + \sum_{h \in H} c_h y_h \tag{1}$$

subject to:

$$x_a \leq q_a \quad \text{for all } a \in A \tag{2}$$

$$\sum_{a \in \delta^+(\{h\})} x_a \leq q_h y_h \quad \text{for all } h \in H \tag{3}$$

$$y_h \in \{0,1\} \quad \text{for all } h \in H, \tag{4}$$

and there exist f_a^k such that

$$x_a = \sum_{k \in K} f_a^k \quad \text{for all } a \in A \tag{5}$$

and for each commodity $k = (i,j)$:

$$\sum_{a \in \delta^+(\{v\})} f_a^k = \sum_{a \in \delta^-(\{v\})} f_a^k \quad \text{for all } v \in V \tag{6}$$

$$f_a^k \geq 0 \quad \text{for all } a \in A \tag{7}$$

$$f_{(j,i)}^k = d_k \tag{8}$$

$$f_{(u,v)}^k = 0 \quad \text{for all } (u,v) \in \overline{K} \setminus \{(j,i)\}. \tag{9}$$

Constraints (6)–(9) require the existence of a flow circulation in G moving exactly d_k units of commodity k. Equalities (5) gather the individual flows into a common one that must satisfy the arc-capacity requirements in (2) and the node-capacity requirements in (3). Finally, (4) impose that a 0-1 decision on the routers must be taken. Then, vector x represents the total communication units circulating through the network, and is a vector of continuous variables, while vector y is a 0-1 vector to decide the routers to be installed. The main difference with a multi-commodity flow formulation problem is the fact that x is not required to be a vector of integer variables. Still, the problem remains $\mathcal{NP}$-hard due to the integrability requirement on the y vector.

Magnanti and Wong [11] observed that it is computationally useful to add the upper bound inequalities $f_a^k \leq q_h y_h$ for all $k \in K$, for all $h \in H$ and for all $a \in \delta^+(\{h\}) \cup \delta^-(\{h\})$, even if they are redundant when inequalities (3) are also present in the model. Surprisingly, this was also confirmed by our experiments.

In addition, we also got better computational results by extending the model with the equations

$$\sum_{a\in\delta^+(\{v\})} x_a = \sum_{a\in\delta^-(\{v\})} x_a \qquad \text{for all } v \in V.$$

At first glance, a simple way to solve the CHP is to provide the Mixed Integer Programming model to a general purpose solver, like `Cplex 8.1`. Nevertheless, this model has the disadvantage of having a large number of continuous variables and constraints, and therefore it is very unlikely to help when solving medium-size instances. The following sections will describe algorithms that decompose this large model into smaller ones to avoid this disadvantage.

3 A First Approach: Dantzig-Wolfe Decomposition

Dantzig and Wolfe [5] propose a decomposition technique to solve large linear programming problems, and this principle can be used to solve model (6)–(9). The basic idea from (5) is that a solution x is a sum of flows, each one associated to a commodity k. A decomposition approach consist in solving iteratively the continuous relaxation of a *master problem* (1)–(5), where for each $k \in K$ the flow variables $[f_a^k : a \in A]$ are replaced by a convex combination of (extreme) solutions of a *subproblem* defined by (6)–(9).

More in detail, the procedure is an iterative procedure. At each iteration we solve a continuous relaxation of a *master problem* defined by an index subset F_k of flows g^l for each commodity $k \in K$, and then we consider a *subproblem* for each commodity k to check if there is a flow with negative reduced cost (defined with the dual variables from the master). The master problem is defined by the objective function (1) and the constraints (2)–(4), plus the additional constraint

$$x_a = \sum_{k\in K}\sum_{l\in F_k} \lambda_l g_a^l$$

$$\sum_{l\in F_k} \lambda_l = 1 \quad \text{for all } k \in K$$

$$\lambda_l \geq 0 \quad \text{for all } l \in F_k \text{ and for all } k \in K,$$

which in a sense replaces (5).

For each commodity, the subproblem is simply a min-cost flow problem (6)–(9). If a flow g^l with negative cost is found then it is included in F_k and another iteration is done. When all flows have non-negative reduced costs for all commodities then the continuous relaxation of the master problem has been optimally solved, but potentially a branch-and-bound process should start to guarantee integrality on the y variables. At each node of the branch-and-bound procedure the same procedure performs using the candidate sets F_k produced at the previous nodes.

A minor modification in the model would allow to replace the min-cost flow computations by shortest path problems. To do that we must simply observe that the min-cost flow problem (6)–(9) has a fix capacity requirement on the arc (j, i) if $k = (i, j) \in K$, while all other arcs have no capacity upper bounds. Then the extreme solutions of this polyhedron are clearly defined by paths from i to j. Arc capacities exist in CHP, but these restrictions are ensured by constrains (2) in the master problem. Then the flows whose existence should be checked are uncapacitated flows, which is equivalent to check, for each commodity $k = (i, j)$, if j is reachable from i in the graph defined by the solution x^* of the master problem. Moreover, when the shortest-path problem replaces the flows, then the modification consists in adding in the master problem the constraint $x_{ji} \geq d_k$ for each commodity $k = (i, j)$.

The full model is then a column-generation approach at each node of a branch-and-bound scheme.

4 A Second Approach: Benders' Decomposition

Another alternative to skip the large number of flow variables in model (1)–(9) is to use Benders' Decomposition (see e.g. Benders [3] and Nemhauser and Wolsey [13]), which is the dual variant of Dantzig-Wolfe Decomposition in Linear Programming. Indeed, a vector $[y_h : h \in H]$ and a vector $[x_a : a \in A]$ satisfying (2)–(4) define a feasible solution for the CHP, if and only if, for each $k \in K$ there exists a vector $[f_a^k : a \in A]$ in the polyhedron defined by (5)–(9). Farkas' Lemma gives a way of imposing the existence of such solutions through linear inequalities as follows. Let us consider a dual variable α_v^k associated to each equation in (6), a dual variable $\beta_{(u,v)}^k$ associated to each equation in (8)–(9), and a dual variable $\gamma_{(u,v)}$ associated to each equation in (5). Then, the polyhedron (5)–(9) is non-empty if and only if all (extreme) rays of the polyhedron cone

$$\begin{aligned} \alpha_u^k - \alpha_v^k + \gamma_{(u,v)} &\geq 0 \quad \text{for all } (u,v) \in A \setminus \overleftarrow{K}, k \in K \\ \alpha_u^k - \alpha_v^k + \beta_{(u,v)}^k + \gamma_{(u,v)} &\geq 0 \quad \text{for all } (u,v) \in \overleftarrow{K}, k \in K \end{aligned}$$

satisfy

$$\sum_{k=(i,j)\in K} d_k \beta_{(j,i)}^k + \sum_{a \in A} x_a \gamma_a \geq 0. \tag{10}$$

Some extreme rays (α, β, γ) of this cone are associated to a subset $S \subseteq V$ and are defined as

$$\alpha_v^k = \begin{cases} +1 & \text{if } v \in S \\ 0 & \text{otherwise,} \end{cases}$$

$$\beta_a^k = \begin{cases} -1 & \text{if } a \in \delta^+(S) \\ 0 & \text{otherwise,} \end{cases}$$

$$\gamma_a = \begin{cases} +1 & \text{if } a \in \delta^-(S) \\ 0 & \text{otherwise.} \end{cases}$$

By replacing these values in (10) we obtain

$$\sum_{a \in \delta^+(S)} x_a \geq \sum_{i \in I \cap S} \sum_{j \in J \setminus S} d_{ij} \tag{11}$$

for all $S \subset V$. Unfortunately these extreme rays do not generate all the cone, and therefore for the moment we do not have an alternative model for the CHP.

Constraints (11) are known in the literature related to the *Multi-commodity flow formulation* (see, e.g., Chapter 6 in [9]) and are termed *Cut Inequalities* [2] or *Bipartition Inequalities* [8]. Unfortunately, no polynomial algorithm is known for separating these inequalities, and the best approach consists in solving a max-cut problem. Still, it is interesting to notice that they are only a subfamily of (10), which can be separated by solving a linear programming problem.

The overall approach is then another iterative procedure where a master problem is solved at each iteration. This master problem is defined by considering the objective function (1) and the constraints (2)–(4), plus a set of inequalities generated by the subproblem. Given a solution x^* of the master problem, at each iteration the subproblem checks if the objective function

$$\min \sum_{k=(i,j)\in K} d_k \beta^k_{(j,i)} + \sum_{a\in A} x^*_a \gamma_a$$

over the above cone is unbounded or not. If a subproblem is unbounded, then the found direction $(\beta^k_{(j,i)}, \gamma_a)$ defines a violated inequality (10) to be added to the new master problem.

A relevant difference when comparing this decomposition approach with the one presented in the previous section is that constraint (5) is part of the master problem when using Dantzig-Wolfe's decomposition, while it goes to the subproblem when using Benders' decomposition.

An observation very useful in practice is based on the fact that for each commodity $k = (i, j)$ only the variables $\beta^k_{(j,i)}$ are necessary in (10), and therefore it is not necessary to overload the subproblem by considering all the variables $\beta^k_{(u,v)}$ for all (u, v) in the cone. This consideration substantially reduces the size of the subproblem solved at each iteration.

5 A Third Approach: Double Decomposition

The approach presented in the previous section has the inconvenience of requiring a large subproblem to be solved at each iteration. A way to avoid this disadvantage consists in applying, once again, a decomposition approach and to solve the subproblem by another iterative procedure similar to the one used with the master problem. More precisely, let us consider the master problem as in the above Benders' decomposition approach. As before, during the iterative procedure a subproblem will generate violated constraints when it is unbounded, until the optimal solution of the subproblem is zero (i.e., it is not unbounded).

Instead of solving the subproblem by using a linear programming solver, we write its dual program and observe that it is a linear program with a very reasonable number of variables and with an exponential number of constraints. The advantage is that not all the constraints are necessary simultaneously, and the most violated one can be found by solving shortest path problems. The details are as follows. Given a solution x^* of the master problem, the associated subproblem consists of checking the feasibility of the linear system of inequalities (5)–(9) for $x = x^*$. By Farkas' lemma, this is equivalent to check the potential unboundness of the objective function

$$\min \sum_{k\in K} d_k \beta^k + \sum_{a\in A} x^*_a \gamma_a$$

over the directions (β, γ) of the cone described by the collection of inequalities:

$$\sum_{a\in A} z'_a \gamma_a + \beta^k \geq 0 \quad \text{for all } k \in K,$$

being z' the characteristic vector of a path going from i to j for each commodity $k = (i,j) \in K$, and by the trivial constraints:

$$\gamma_a \geq 0 \quad \text{for all } a \in A$$
$$\beta^k \leq 0 \quad \text{for all } k \in K.$$

We are reducing the feasible region of this problem by considering signs on the variables, based on the properties that one can replace —without loss of generality— all the equalities in (6) by inequalities of type "$\leq$", and all the equalities in (8) by inequalities of type "$\geq$".

Observe that one can expect the variables z' to define flows from i to j instead of paths, but only the extreme flows are necessary to generate all of them and the subproblem has no capacities on the arcs, as mentioned at the end of Section 3. Moreover, once again, for each commodity $k = (i,j) \in K$ one could expect to have variables $\beta^k_{(u,v)}$ for all (u,v), but only the one associated to arc (j,i) is necessary if the other arcs $(u,v) \in \overleftarrow{K}$ are removed when generating the paths.

Finally, even in this dual version, the linear program is difficult to solve if all the constraints are given to a linear programming solver. The advantage of this approach, whereas, is that not all the constraints are necessary simultaneously and the most violated one (if any) by a given solution (β^*, γ^*) can be generated by solving shortest path problems. Observe that for each commodity $k = (i,j)$, z' is a path going from i to j. More precisely, for each commodity $k \in K$, let us compute the shortest path on the graph $G^k = (\{i,j\} \cup H, A \setminus \overleftarrow{K})$ where each arc a is associated to a cost γ^*_a. Let z' be the characteristic vector defined by setting z'_a to 1 if arc a is in the path used by the commodity k, and to 0 in another case. If $\sum_{a\in A} \gamma^*_a z'_a + {\beta^*}^k < 0$, then the vector z' generates a violated constraint to be added to the subproblem, that must be reoptimized. When z' does not induce a violated inequality for any of the commodities, then the subproblem was solved, and the last unbounded direction (β^*, γ^*) defines a violated constraint for the master problem. The iterative procedure on the master problem stops when the subproblem finds the zero vector as the optimal solution.

Once again, this mechanism allows to solve the continuous relaxation of the master problem, hence a branch-and-bound procedure must start if a y_h variable has a fractional value. At each node of the branch-and-bound procedure the same iterative procedure can be applied.

During the process three programs are solved: a master problem by using a linear programming solver, a subproblem by a linear programming solver, and a shortest path problem by an specialized algorithm. An advantage of the process is that during the different iterations the region of the three solved programs remains the same. Indeed, all the cuts generated for the master problem and for the subproblems can be used in the next resolutions. Also the shortest path problems can be always computed on the same graphs. The difference between iterations consists only in the different coefficients in the objective function.

6 Computational Experiments

The decomposition techniques described in Sections 3 and 4 have similar performances, while the one in Section 5 involves a more elaborated way of solving the subproblem. This section shows the results of experiments conducted on solving some random instances with our implementations of the decomposition approaches in Sections 4 and 5. These experiments were carried out on a Pentium IV 1500 Mhz. using the branch-and-cut framework of `CPLEX 8.1`.

The instance generator is next described. Given the sets *I, J, H,* the arc density of the graph induced by *H* was fixed to a parameter taking values in $\{30\%, 50\%, 85\%\}$, which gave instances with low, medium and high density. The percentage of arcs from $(I \times H) \cup (H \times J)$ was fixed to 80%. These settings were chosen in order to likely produce instances with feasible solutions. The amount of information d_{ij} from $i \in I$ to $j \in J$ was generated in [1, 5]. The capacities q_h and q_a were generated in $[1, |I \times J| \cdot 2, 5]$. The costs c_h and c_a were generated in [1, 50]. We considered $(|I|, |J|, |H|)$ in $\{(2,2,4), (3,3,5), (5,5,5), (5,5,10)\}$, and for each triplet we generated five random instances with the above features.

Table 1 shows average results of applying the two decomposition approaches on feasible instances. Column headings display:

– The density (d) of the generated graphs: low (l), medium (m) or high (h).
– The percentage ratio *LB/opt,* where *LB* is the objective value of the LP-relaxation computed at the root node of the branch-and-cut tree, and *opt* is the optimal CHP solution value.
– The time in seconds spent in the root node (r-time), number of nodes in the branch-and-cut tree (nodes), total computing time (time) and separated inequalities (cuts) used by the Benders' decomposition algorithm.
– The time in seconds spent in the root node (r-time'), number of nodes in the branch-and-cut tree (nodes'), total computing time (time'), inequalities introduced in the master problem (cuts'), and inequalities introduced in the subproblems (cuts") while applying the Double Decomposition algorithm.

The table clearly shows that the Double Decomposition algorithm outperforms the classical Benders' approach. Indeed, in our experiments it was about ten times faster on the largest instances. This is explained by the fact that the Double Decomposition exploits the combinatorial structure of the subproblem, while Benders' Decomposition solves it as a standard linear program.

In both approaches, the number of inequalities reported in the column *cuts* refers to the cuts added to the master problem, which is slightly smaller than the number of subproblems solved. In the Benders' approach the subproblems are solved as black boxes by an LP solver, while in the Double Decomposition each subproblem requires to solve a sequence of shortest path problems. The number of path problems is slightly larger than the number of cuts added to the subproblem, which tends to be a very small number. Column *cuts"* in Table 1 shows the total number of inequalities added to all the subproblems while, on average, the number of shortest path inequalities added to each subproblem varies between 4 for the smallest instances and 12 for the largest ones. This explains the better performance of the new technique.

Table 1. Results on random instances

			Benders' Decomp.				Double Decomp.				
$(\|I\|,\|J\|,\|H\|)$	d	LB/opt	r-time	nodes	time	cuts	r-time'	nodes'	time'	cuts'	cuts"
(2,2,4)	l	98,0	0,1	1,6	0,1	1,4	0,1	1,6	0,1	1,4	11,4
	m	99,8	0,1	1,2	0,1	1,2	0,1	1,2	0,1	2,2	13,4
	h	99,6	0,1	1,2	0,1	1,4	0,2	1,2	0,2	1,4	14,8
(3,3,5)	l	99,0	0,2	2,5	0,2	4,8	0,3	2,8	0,3	5,5	35,3
	m	99,8	0,3	3,4	0,3	7,4	0,3	2,8	0,3	6,0	39,6
	h	96,9	0,3	3,4	0,3	7,6	0,5	3,4	0,5	7,4	63,0
(5,5,5)	l	99,6	18,1	1,8	18,4	25,8	1,6	1,8	1,6	24,8	92,0
	m	99,6	18,8	1,5	18,8	25,8	1,9	1,5	2,0	24,8	114,8
	h	99,6	21,1	2,3	21,1	23,8	2,5	3,2	2,5	23,0	155,8
(5,5,10)	l	94,9	123,4	12,8	148,6	65,2	10,6	15,6	12,2	69,0	394,0
	m	95,1	176,2	13,4	219,1	72,2	16,1	13,4	18,4	64,4	611,8
	h	96,3	225,2	12,0	297,7	66,8	29,0	11,8	32,3	62,0	1144,6

7 Conclusions

We have introduced the combinatorial optimization problem of deciding the hubs to be opened in a telecommunications network to send given demands of information from source terminals to destination terminals at minimum cost. We are not assuming that all the hubs are connected by an arcs, as happens in works related with hub location in the literature. Capacities on the arcs and hubs are considered, and therefore the problem is referred as *Capacitated Hub Problem.*

This problem is very closely related to the well-known Multi-commodity Flow Problem in Network Design, but differs in the fact that our objective function is not piece-wise in the total flow traversing an arc. The complexity in our problem is due to the cost for opening hubs, thus managing a zero-one variable for each node.

This paper presents a Mixed Linear Programming model. The large numbers of continuous variables and constraints create difficulties when applying a general purpose solver like `Cplex 8.1`, and therefore we have presented different approaches to solve it by decomposition techniques. The first two approaches are based on single decomposition techniques, while the third one applies a two-level nested decomposition scheme.

Acknowledgements

This work has been partially supported by "Ministerio de Ciencia y Tecnología", Spain (research project TIC-2003-05982-C05-02).

References

1. M. Almiñana, L.F. Escudero, J.F. Monge, J. Sánchez-Soriano, "On solving the Enrouting Protocol Problem under uncertainty", proceedings of the CORAL meeting, http://webpages.ull.es/users/saderyl/Coral.htm
2. F. Barahona, "Network Design using cut inequalities", *SIAM Journal on Optimization* 6 (1996) 823–837.
3. J.F. Benders, "Partitioning Procedures for Solving Mixed Variables Programming Problems", *Numerische Matheamtik* 4 (1962) 238–252.
4. J.F. Campbell, "Integer programming formulations of discrete hub location problems", *European Journal of Operational Research* 72 (1994) 387–405.
5. G.B. Dantzig, P. Wolfe, "Decomposition Principle for Linar Program", *Operations Research* 8 (1960) 101–111.
6. J. Ebery, M. Krishnamoorthy, A. Ernst, N. Boland, "The capacitated multiple allocation hub location problem: Formulations and algorithms", *European Journal of Operational Research* 120 (2000) 614–631.
7. E. Ernst, M. Krishnamoorthy, "Exact and heuristic algorithms for the uncapacitated multiple allocation p-hub problem", *European Journal of Operational Research* 104 (1998) 100–112.
8. V. Gabrel, A. Knippel, M. Minoux, "Exact solution of multicommodity network optimization problems with general step cost functions", *Operations Research Letters* 25 (1999) 15–23.
9. M. Gondran, M. Minoux. Graphs and algorithms. John Wiley and Sons, 1984.
10. K. Holmberg, D. Yuan, "A Langrangian heuristic based branch-and-bound approach for the capacitated network design problem", *Operations Research* 48 (2000) 461–481.
11. T.L. Magnanti, R.T. Wong "Network design and transportation planning: models and algorithms", *Transportation Science* 18 (1984) 1–55.
12. G. Mayer, B. Wagner, "HubLocator: an exact solution method for the multiple allocation hub location problem", *Computer & Operations Research* 29 (2002) 715–739.
13. G.L. Nemhauser, L.A. Wolsey, *Integer and Combinatorial Optimization,* Wiley-Interscience, 1988.
14. M. O'Kelly, D. Bryan, D. Skorin-Kapov, J. Skorin-Kapov, "Hub network design with single and multiple allocation: A computational study", *Location Science* 4 (1996) 125–138.
15. D. Skorin-Kapov, J. Skorin-Kapov, M. O'Kelly, "Tight linear programming relaxations of uncapacitated p-hub median problems", *European Journal of Operational Research* 94 (1996) 582–593.

An Efficient Method to Schedule New Trains on a Heavily Loaded Railway Network

Laura Ingolotti[1], Federico Barber[1], Pilar Tormos[2], Antonio Lova[2], M.A. Salido[3], and M. Abril[1]

[1] DSIC, Universidad Politecnica de Valencia, Spain
{lingolotti, fbarber, mabril}@dsic.upv.es
[2] DEIOAC, Universidad Politecnica de Valencia, Spain
{ptormos, allova}@eio.upv.es
[3] DCCIA, Universidad de Alicante, Spain
masalido@dccia.ua.es

Abstract. With the aim of supporting the process of adapting railway infrastructure to present and future traffic needs, we have developed a method to build train timetables efficiently. In this work, we describe the problem in terms of constraints derived from railway infrastructure, user requirements and traffic constraints, and we propose a method to solve it efficiently. This method carries out the search by assigning values to variables in a given order and verifying the satisfaction of constraints where these are involved. When a constraint is not satisfied, a guided backtracking is done. The technique reduces the search space allowing us to solve real and complex problems efficiently.

1 Introduction

The train scheduling problem is basically an optimization problem which is computationally difficult to solve. Several models and methods have been analyzed to solve it [1], [2]. In our method, we consider a heterogeneous railway network and we focus on adding new trains to a railway network that is already occupied by trains in circulation. The models proposed above are not efficient for the type of problem that we are considering. The majority of the papers published in the area of periodic timetabling in the last decade are based on the Periodic Event Scheduling Problem (PESP) introduced by Serafini and Ukovich [8]. Specifically, an efficient model that uses the PESP and the concept of symmetry is proposed in [6]. However, we cannot use the concept of symmetry because: (i) we allow different types of trains and this does not guarantees the necessary symmetry to be able to use these models; and (ii) the use of the infrastructure may not be symmetrical. There are related works to railway problems such as the following: allocation of n trains in a station minimizing the number of used tracks and allowing the departure of trains in the correct order [4], allocation of new stations through the railway network to increase the number of users [7], etc. There are tools to solve certain kinds of problems such as the Rescheduling tool [3] or the

TUFF scheduler [5]. The Rescheduling tool allows the user to modify a timetable when trains in a section of track cannot run according to the infrastructure, ensuring that scheduling rules are not violated. The TUFF scheduler describes a constraint model and solver for scheduling trains on a network of single tracks used in both directions. However, our problem differs from the type of problems dealt with by the methods mentioned above. In the following section, we explain in more detail the type of problem that we have dealt with.

2 Problem Specification

We propose to add new trains on a heterogeneous heavily loaded railway network, minimizing the traversal time of each new train. The timetables for the new trains are obtained in a search space that is limited by traffic constraints, user requirements, railway infrastructure and network occupation. The problem specification does not require that all considered trains visit the same sequence of locations. There may be many types of trains, which implies different: velocities, safety margins, commercial stops and journeys. Our method takes into account the following scenario to generate the timetables corresponding to the new trains:

1. two sets of ordered locations $L_D = \{l_k, l_{k+1}, ..., l_{k+m}\}$ and $L_U = \{l_h, l_{h+1}, ..., l_{h+n}\}$, such that $\{\exists i, j \backslash l_i \in L_D \wedge l_{i+1} \in L_D \wedge l_j \in L_U \wedge l_{j+1} \in L_U \wedge l_i = l_{j+1} \wedge l_{i+1} = l_j\}$. A pair of adjacent locations can be joined by a single or double track section. L_U and L_D correspond to the journey of trains going in up and down directions, respectively.
2. a set of trains for each direction. $T_D = \{t_0, t_2, ..., t_d\}$ is the set of trains that visit the locations in L_D in the same order given by this sequence and going in the *down direction.* $T_U = \{t_1, t_3, ..., t_u\}$ is the set of trains that visit the locations in L_U in the same order given by this sequence and going in the *up direction.* The subscript i in the variable t_i indicates the departure order among the new trains going in the same direction.
3. a journey for each set of trains (T_U and T_D) specifies the traversal time for each section of track in L_D and in L_U ($\mathrm{R}_{i \to i+1}$), and the minimum stop time (S_i) for commercial purposes in each l_i.

Considering $\mathrm{t}_y\mathrm{dep\_l}_x$ and $\mathrm{t}_y\mathrm{arriv\_l}_x$ as the departure and arrival times of train t_y from/at location l_x, the problem consists of finding the running map that minimizes the average traversal time, satisfying all the following constraints:

– *Initial Departure Time.* The first train must leave from the first station of its journey within a given time interval ($[min_D, max_D]$ for trains going in the down direction and $[min_U, max_U]$ for trains going in the up direction).

$$min_D \leq t_0 dep\_l_k \leq max_D \wedge min_U \leq t_1 dep\_l_h \leq max_U \ . \qquad (1)$$

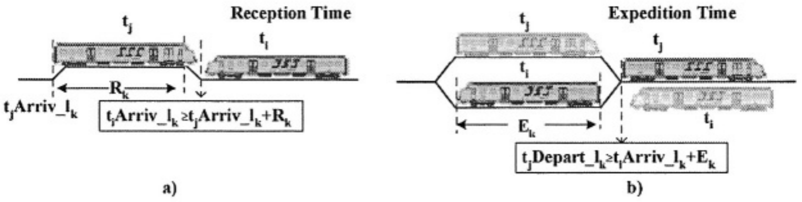

Reception Time
t_j
t_i
R_k
t_jArriv_l_k
t_iArriv_$l_k \geq t_j$Arriv_l_k+R_k
a)
Expedition Time
t_j
t_i
E_k
t_jDepart_$l_k \geq t_i$Arriv_l_k+E_k
b)

Fig. 1. Reception and Expedition time constraint

– *Frequency of Departure.* It specifies the period (F_U/F_D) between departure of two consecutive trains in each direction from the same location,

$$\{\forall t_i, l_j \backslash t_i \in \{T_D - \{t_d\}\} \wedge l_j \in \{L - \{l_{k+m}\}\}\}, t_{i+2}dep\_l_j = t_i dep\_l_j + F_D\ . \quad (2)$$

$$\{\forall t_i, l_j \backslash t_i \in \{T_U - \{t_u\}\} \wedge l_j \in \{L - \{l_{h+n}\}\}\}, t_{i+2}dep\_l_j = t_i dep\_l_j + F_U\ . \quad (3)$$

– *Minimum Stops.* A train must stay in a location l_j at least S_j time units,

$$\{\forall t_i, l_j \backslash t_i \in \{T_D \cup T_U\} \wedge l_j \in \{L_D \cup L_U - \{l_k, l_h, l_{k+m}, l_{h+n}\}\}\},$$
$$t_i dep\_l_j \geq t_i arriv\_l_j + S_j\ . \quad (4)$$

– *Exclusiveness.* A single track section must be occupied by only one train at the same time.

$$\{\forall t_j, t_i, l_x, l_y / t_j \in T_D \wedge t_i \in T_U \wedge l_x \in L_D - \{l_{k+m}\} \wedge$$
$$l_y \in L_U - \{l_h\} \wedge l_x = l_{y+1} \wedge l_{x+1} = l_y\},$$
$$t_i dep\_l_y \geq t_j arriv\_l_y \vee t_j dep\_l_x \geq t_i arriv\_l_x\ . \quad (5)$$

– *Reception Time.* At least R_x time units are required at location l_x between the arrival times of two trains going in opposite directions (Figure 1.a).

$$\{\forall t_j, t_i, l_x / t_j \in T_D \wedge t_i \in T_U \wedge l_x \in \{L_D - \{l_k\} \cap L_U - \{l_h\}\}\},$$
$$t_j arriv\_l_x \geq t_i arriv\_l_x + R_x \vee t_i arriv\_l_x \geq t_j arriv\_l_x + R_x\ . \quad (6)$$

– *Expedition Time.* At least E_x time units are required at location l_x between the arrival and departure times of two trains going in opposite directions (Figure 1.b).

$$\{\forall t_j, t_i, l_x / t_j \in T_D \wedge t_i \in T_U \wedge l_x \in \{L_D - \{l_{k+m}\} \cap L_U - \{l_{h+n}\}\}\},$$
$$t_j dep\_l_x \geq t_i arriv\_l_x + E_x \vee t_i dep\_l_x \geq t_j arriv\_l_x + E_x\ . \quad (7)$$

– *Precedence Constraint.* Each train employs a given time interval $(R_{x \to x+1})$ to traverse each section of track $(l_x \to l_{x+1})$in each direction.

$$\{\forall t_i, t_j, l_x, l_y / t_i \in T_D \wedge t_j \in T_U \wedge l_x \in \{L_D - \{l_{k+m}\}\} \wedge l_y \in \{L_U - \{l_{h+n}\}\}\},$$
$$t_i arriv\_l_{x+1} = t_i dep\_l_x + R_{x \to x+1} \wedge\ t_j arriv\_l_{y+1} = t_j dep\_l_y + R_{y \to y+1}. \quad (8)$$

– *Capacity of Each Station.* The number of trains that may stay simultaneously in a station depends on the number of available tracks in it.
– *Closure Times.* Traffic operations and/or passing of trains are not allowed at the closing times of a station.

3 Sequential Algorithm

In this section, we explain the algorithm that is used to solve the specified problem in Section 2. We have named it *"Sequential"* because of the way that it generates the timetable for each new train (Figure 2). For each iteration, the Sequential algorithm constitutes a subset of the whole search space where it searches the values for the problem variables that satisfy all the problem constraints (Section 2). The assignment of valid values to the problem variables generates a timetable (if there is a feasible solution in the subset) for each new train (line 08-14 in Figure 2). The elements of the reduced search space depend on the following values:

1. *Initial departure time* for the first train going in the *down* direction (init_dep_down in Figure 2). A value belonging to the time interval $[min_D,max_D]$ is chosen randomly at each iteration (Constraint 1 in Section 2). This time interval is part of the input parameters *I* (one of the input parameters of the Sequential Algorithm in Figure 2) given by the final user.
2. *Initial departure time* for the first train going in the *up* direction (init_dep_up in Figure 2). A value belonging to the time interval $[min_U,max_U]$ is chosen randomly at each iteration (Constraint 1 in Section 2). This time interval is part of the input parameters *I* (one of the input parameters of the Sequential Algorithm in Figure 2) given by the final user.

```
01    procedure Sequential_Algorithm(I,C)
02    begin
03       while(Enough_Time())
04          S=Generate_Set_Ref_Station()
05          ref_st=Get_Ref_Station(S)
06          init_dep_down=Get_Init_Dep_Time($min_D$,$max_D$)
07          init_dep_up=Get_Init_Dep_Time($min_U$,$max_U$)
08          sched1=Get_Partial_Sched(init_dep_down,$l_k$,ref_st,$T_D$)
09          sched2=Get_Partial_Sched(init_dep_up,$l_h$,ref_st,$T_U$)
10          init_dep_down=$t_0$arriv_$l_{ref\_st}$+$S_{ref\_st}$
11          init_dep_up=$t_1$arriv_$l_{ref\_st}$+$S_{ref\_st}$
12          sched3=Get_Partial_Sched(init_dep_down,ref_st,$l_{k+m}$,$T_D$)
13          sched4=Get_Partial_Sched(init_dep_up,ref_st,$l_{h+n}$,$T_U$)
14          new_sched=sched1+sched2+sched3+sched4
15          if(Is_Better(new_sched,best_sched))
16             best_sched=new_sched
17       end while
18       Show(best_sched)
19    end
```

Fig. 2. Sequential Algorithm

3. *Reference Station* (ref_st in Figure 2). When the assigned value to a problem variable that represents the departure time of a train violates Constraint 5 defined in Section 2 (to avoid crossings between trains going in opposite directions), the process must decide which of the two trains will have to wait for the section track release. The decision taken by the process will state a priority order between the trains competing for the same resource, the single track section.

   When the two trains competing for a single track section are a train in circulation and a new train that should be added to the railway network, the priority order will always be the same. The new train will have to wait until the train in circulation releases the single track section.

   When the two trains competing for a single track section are both new trains, the priority order is decided according to the position of the single track section with respect to one station, which we name the *reference station.* The *reference station* divides the journey of each new train into two parts: the first part goes from the initial station of the journey to the *reference station;* the second part goes from the *reference station* to the last station of the journey. Each train will have priority on the single track sections that belong to the first part of its journey. In Figure 3, S_2 is the reference station and it divides the journey of each new train into two parts. For the trains going in the *down* direction (t_0, t_2 and t_4), the first part of their journey is composed of the track sections (S_0-S_1) and (S_1-S_2); the second part of their journey is composed of (S_2-S_3) and (S_3-S_4). For the trains going in the *up* direction (t_1, t_3 and t_5), the first part of their journey is composed of (S_4-S_3) and (S_3-S_2); the second part of their journey is composed of (S_2-S_1) and (S_1-S_0). In Figure 3, the possible crossing between the trains t_0 and t_1 (in case that t_0 left from S_2 as soon as possible) is highlighted by a circle. The single track section (S_2-S_3) belongs to the first part of the journey of t_1, and therefore this train has greater priority than t_0 on this track section. Then, t_0 will have to wait until the single track section (S_2-S_3) is released by the train t_1. The dotted lines represent the position of the train t_0 if it had left from the station S_2 as soon as possible. The continuous lines represent

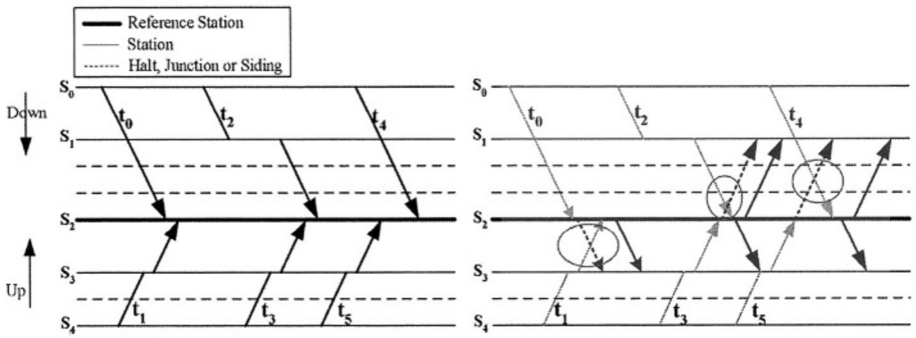


Fig. 3. Priority Order between new trains defined by a Reference Station

the real departure time of the train t_0 after the single track section had been released by t_1. The same reasoning is applied to the other cases in Figure 3 and are designated by circles.

The sequential algorithm iterates obtaining new solutions until the time given by the user has been completely used up or until the user interrupts the execution (line 03 in Figure 2). For each iteration, the sequential algorithm compares the obtained scheduling with the best scheduling obtained to that point. The scheduling that produces the least average traversal time for each new train is the best scheduling. The function *Is_Better (new_timetable, best_timetable)* returns TRUE if the new scheduling *(new_sched* in Figure 2, obtained in the current iteration) is better than the best scheduling (*best_sched* in Figure 2, obtained until that time). Finally, the sequential algorithm returns the best timetable that has been obtained during the running time.

We have decided not use general solvers such as LINGO or CPLEX because we needed integer binary variables to model given constraints in a mathematical model and the resulting complexity was too high for real world problems. For this reason, we implement a custom solver, which uses the knowledge about this type of problem to solve it in a more efficient way.

3.1 Description of an Iteration of the Sequential Algorithm

At each iteration, the sequential algorithm generates a complete scheduling for each group of trains (T_D and T_U) in two steps. In Figure 2, *sched1* and *sched2* correspond to the scheduling generated for the first part of the journey of each new train going in the *down* and the *up* direction, respectively. In Figure 2, *sched3* and *sched4* correspond to the scheduling generated for the second part of the journey of each new train going in the *down* and the *up* direction, respectively. The complete scheduling is generated in this way in order to establish greater priority for each new train on the first part of its journey. In the case of there being the possibility of a crossing in a track section, the greatest priority will be given to the new train whose timetable had been assigned first on that track section. The first and second part of a journey are established by the chosen reference station at the current iteration. Figure 4 shows how the scheduling for one part of the whole journey is generated. *Verify_Constraints*(*st, next_st, dep_time,* t_i) verifies that all problem constraints are satisfied by the train t_i in the track section limited by the stations *st* and *next_st*. This function returns the time that must be added to the departure time of t_i in order to satisfy the violated constraint. If the function returns 0, then no constraint has been violated. Consider the set $L' = \{l_x, l_{x+1}, ..., l_{x+p}\}$ as the ordered set of locations visited by the train t_i, from $l_x = st$ to $l_{x+p} = next\_st$. This function assigns values to the variables $ti\_dep\_l_j$ and $ti\_arriv\_l_h$ such that $x <= j < x+p$ and $x < h <= x+p$ (Figure 5). Figure 5 shows an example of how the constraint that avoids a crossing between trains going in opposite directions is verified. Consider that train t_2 is the train t_i (the train whose timetable is being created), and t' is the train whose timetable has been created first and cannot be modified. The initial value assigned as departure time from st=l_x to t_2 causes a crossing of t' with t_2 (see

```
procedure Get_Partial_Sched(init_dep,first_st,last_st,T)
begin
   st=first_st
   dep_time=init_dep
   while(st != last_st)
      next_st=Get_Next_St(st)
      i=Get_First_Train(T)
      while($t_i$!=NULL)
         error=Verify_Constraints(st,next_st,dep_time,$t_i$)
         if(error>0)
            if(Is_Required_Frequency())
               i=Get_First_Train(T)
               $t_i$dep_$l_{st}$=$t_i$dep_$l_{st}$+error
            end if
         else
            if(Is_Required_Frequency())
               dep_time=$t_i$dep_$l_{st}$+$F_T$
            else
               dep_time=$t_{i+2}$arriv_$l_{st}$+$S_T$
            end if
            i=i+2
         end if
      end while
      st=next_st
   end while
end
```

Fig. 4. Partial Scheduling

```
int function Verify_Constraints(st,next_st,dep_time,$t_i$)
begin
   k=st
   m=next_st
   trav_time=Get_Traversal_Time(st,next_st)
   $t_i$dep_$l_k$=dep_time
   $t_i$arriv_$l_m$=dep_time+trav_time
   error=Verify_Crossing($t_i$dep_$l_k$,$t_i$arriv_$l_h$)
   if(error=0)
      error=Verify_Overtaking($t_i$dep_$l_k$,$t_i$arriv_$l_h$)
      if(error=0)
         error=Verify_Availability_Tracks($t_i$arriv_$l_h$)
         if(error=0)
            error=Verify_Closure_Time($t_i$arriv_$l_h$)
            if(error=0)
               Set_Timetable(st,next_st)
         end if
      end if
   end if
   return error
end
```

Fig. 5. Constraints Verification and Timetable Assignment

the picture on the left in Figure 6). The function *Verify_Constraints* computes the time that must be added to *dep_time* in order to avoid this crossing, and

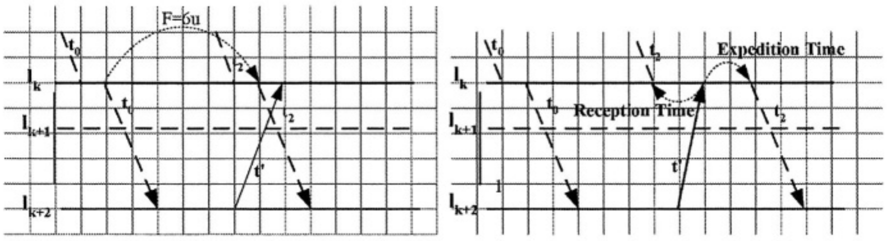


Fig. 6. A crossing detected in a section of track

this value is returned by the function. The picture on the right in Figure 6 shows the necessary delay in the departure time of t_2 from S_2.

The precedence constraint (Constraint 8 in Section 2) is used to propagate the values among the variables*(trav_time* in Figure 5 is the time spent to go from *st* to *next_st)*. Minimum stops constraint(Constraint 4) is used to compute the departure time of a train from its arrival time at the same location(S_T in Figure 4). *Get_Next_Station(st,T)* returns the next station to *st* in the journey corresponding to trains belonging to *T*. *Get-First-Train(T)* returns the first train that starts the journey corresponding to trains belonging to *T*. *Is_Required_Frequency(st)* returns TRUE if all the trains must keep the same departure frequency in the station *st*.

4 Computational Results

The algorithm just described was implemented in C++ and tested on a set of instances from the National Network of Spanish Railways(RENFE), which has collaborated closely with the research group. We have used an Intel Pentium 4 1,6 GHz processor to test the algorithm. Figure 7 shows the different average traversal times obtained by our algorithm as the time elapses for two different instances of the problem specified in Section 2. The picture on the left in Figure 7 shows the different results for the problem of adding 10 new trains in each direction to the railway network occupied by 81 trains in circulation. The main journey of this network is Madrid-Zaragoza (65 locations), but there are different types of trains with different journeys, which are subsets of the main journey. The first new train must leave within the time interval [7:00:00,8:30:00] and the departure frequency between trains in the same direction must belong to the time interval [0:10:00,0:50:00]. The picture on the right in Figure 7 corresponds to the same problem but without taking into account trains in circulation on the railway network. We have not been able to obtain results of similar methods for comparison with our algorithm. Therefore, we have modeled our problem to solve it using LINGO and CPLEX. The models solved by these general solvers do not take into account constraints about the number of tracks and the closure time in each station. The instance tested with these solvers does not take into account trains in circulation because the problem becomes untractable for them. The se-

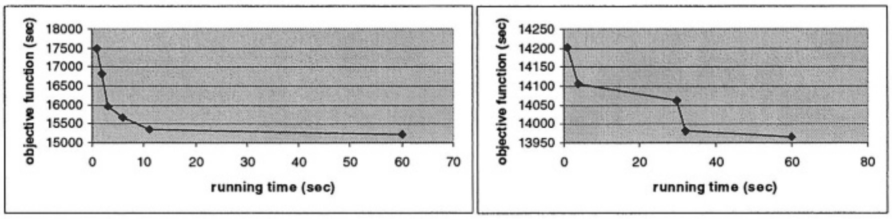


Fig. 7. Average traversal time in function of the running time

quential algorithm takes into account all the problem constraints. It is important to note that LINGO and CPLEX receive the whole search space generated by each problem without any pre-processing to perform some reduction on it. Table 1 has two columns for each solver, the column labeled "Journey Time", and the column labeled "Running Time". The "Journey Time" column shows the objective function value depending on the elapsed time, and the "Running Time" column shows the elapsed time with respect to the start time of each solver. How the objective function value is improved as the elapsed time increases is shown for each solver. The results shown in Table 1 correspond to the following instance: number of new trains for each direction: 10, number of locations that are visited by each train: 42 (Vigo-A Coruna), frequency of departure: 2:00:00, initial departure time interval: [7:00:00,8:30:00].

5 Conclusions

The problem variables are ordered implicitly in the sequential algorithm. The timetable for t_i is assigned before the timetable for t_{i+1} is assigned. The timetable at the location l_x is established before the timetable at l_{x+1} is established. With this ordering, the required backtracking undoes the least number of instantiated variables. In each iteration, the variable values of the problem are computed from an initial departure time($t_0 dep\_l_k = init\_dep\_down$ and $t_1 dep\_l_h = init\_dep\_up$). When the initial departure time is modified, a different subset of the whole search space is obtained, and therefore a different scheduling can be generated. The reference station establishes the priority order between two trains competing for the same resource, a single track section. When the value for this parameter is modified the priority order between trains is modified as well. Therefore, a

Table 1. Running time to obtain each timetable with different solvers

LINGO		CPLEX		SEQUENTIAL	
running time	journey time	*running time*	journey time	*running time*	journey time
120"	2:38:13	15"	2:27:56	<1"	2:28:08
160"	2:26:27	60"	2:23:22	1"	2:25:50
865"	2:21:46	120"	2:21:46	3"	2:21:46

train that in a previous iteration had to wait for the release of a track section, could be the train with greater priority in another iteration. Then, its timetable could be different. Each iteration with a different combination of values for these parameters reduces the search space to a different subset. Each iteration is solved quickly because it only has to explore a reduced part of the whole search space.

Our heuristic consists of obtaining several solutions, where each one is obtained from a different subset of the same search space. This is done instead of searching one single solution in a whole search space, which in real cases may be untractable. The sequential algorithm indicates an order among the problem variables. This order implicitly points out the priority of one train for a resource (platform in a station or single track section). A new order (and a new solution) is produced for each iteration. This is the method that we propose for solving a CSP for this type of problem.

References

1. Bussieck, M.R., Winter, T., and Zimmermann, U.T., 'Discrete optimization in public rail transport', *Math. Programming,* **79(1-3)**, 415–444, (1997).
2. Cordeau, J.F., Toth, P., and Vigo, D., 'A survey of optimization models for train routing and scheduling', *Transportation Science,* **32(4)**, 380–404, (1998).
3. C. K. Chiu, C.M. Chou, J.H.M. Lee, H.F. Leung, and Y.W. Leung, 'A constraint based interactive train rescheduling tool', *Constraints,* **7**, 167–198, (2002).
4. L. Koci and Di Stefano G., 'A graph theoretical approach to shunting problems', *Proceedings of ATMOS 2003 Algorithmic Methods and Models for Optimization of Railways, Electronic Notes in Theoretical Computer Science,Elsevier 92,* **1**, (2003).
5. P. Kreuger, J.O. Carlsson, T. Sjoland, and E. Astrom, 'The tuff train scheduler', *German Puebla, editor, Proceedings of the workshop on Tools and Environments for (Constraints) Logic Programming at the International Logic Programming Symposium ILPS'97,* (1997).
6. C. Liebchen, 'Symmetry for periodic railway timetables', *Proceedings of ATMOS 2003 Algorithmic Methods and Models for Optimization of Railways, Electronic Notes in Theoretical Computer Science 92,* **1**, (2003).
7. M.F. Mammana, S. Mecke, and D. Wagner, 'The station location problem on two intersecting lines', *Proceedings of ATMOS 2003 Algorithmic Methods and Models for Optimization of Railways, Electronic Notes in Theoretical Computer Science,Elsevier 92,* **1**, (2003).
8. Serafini, P., and Ukovich, W., 'A mathematical model for periodic scheduling problems', *SIAM Journal on Discrete Mathematics,* **2(4)**, 550–581, (1989).

Studs, Seeds and Immigrants in Evolutionary Algorithms for Unrestricted Parallel Machine Scheduling

E. Ferretti, S. Esquivel, and R. Gallard

Laboratorio de Investigación y Desarrollo en Inteligencia Computacional,
Universidad Nacional de San Luis, Argentina
{ferretti, esquivel}@unsl.edu.ar

Abstract. Parallel machine scheduling, involves the allocation of jobs to the system resources (a bank of machines in parallel). A basic model consisting of *m* machines and *n* jobs is the foundation of more complex models. Here, jobs are allocated according to resource availability following some allocation rule. In the specialised literature, minimisation of the makespan has been extensively approached and benchmarks can be easily found. This is not the case for other important objectives such as the due date related objectives. To solve the unrestricted parallel machine scheduling problem, this paper proposes MCMP-SRI and MCMP-SRSI, which are two multirecombination schemes that combine studs, random and seed immigrants. Evidence of the improved behaviour of the EAs when inserting problem-specific knowledge with respect to SCPC (an EA without multirecombination) is provided. Experiments and results are discussed.

1 Introduction

Unrestricted parallel machine scheduling problems are common in production systems. The completion time of the last job to leave the system, known as makespan ($\mathbf{C_{max}}$), is one of the most important objective functions to be minimised, because it usually implies high utilization of resources. In a production system it is also usual to stress minimisation of due date based objectives such as average tardiness ($\mathbf{T_{avg}}$), weighted tardiness ($\mathbf{T_{wt}}$), or weighted number of tardy jobs ($\mathbf{N_{wt}}$).

Branch and Bound and other partial enumeration based methods, which guarantee exact solutions, are prohibitively time consuming even with only 20 jobs. To provide reasonably good solutions in very short time the scheduling literature offers a set of dispatching rules and heuristics. Depending on the particular instance of the problem we are facing, some heuristics behave better than others. Among other heuristics [12], evolutionary algorithms (EAs) have been successfully applied to solve scheduling problems [9-11]. Current trends in evolutionary algorithms make use of multiparent [3-5] and multirecombined approaches [6-8]. We call to this latter approach, *multiple crossovers on multiple parents* (MCMP). Instead of applying crossover once on a pair of parents, this scheme applies n_1 crossover operations on a set of n_2 parents. In order to improve the trade-off between exploration and exploitation in the search process a variant called MCMP-SRI [13,14] recombines a breeding individual (stud) by repeatedly mating individuals that randomly immigrate to a mating pool. Under this approach the random immigrants and multi-mating operation with the stud incorporate exploration and exploitation, respectively in the search process.

If we are trying to incorporate knowledge to the blind evolutionary search process, the main issue here is how to introduce problem-specific knowledge? If optimality conditions for the solutions are known in advance we can restrict the search operating only on solutions which hold these conditions. When optimality conditions are unknown, which is the case here, one of the options is to import this knowledge from solutions that come out of heuristics specifically designed for the problem under consideration. These types of knowledge-based intermediate solutions contain some of the features included in the best (optimal or quasi-optimal) solution at the end of the evolutionary process.

Consequently MCMP-SRSI, a latest variant of MCMP-SRI, considers the inclusion of a stud-breeding individual in a pool of random and seed-immigrant parents. Here the seeds generated by conventional heuristics introduce the problem-specific knowledge. The following sections describe the above mentioned scheduling problems, ways of inserting *problem-specific knowledge* and provide a discussion of the results obtained.

2 Scheduling Problems

The problems we are facing [16] can be stated as follows: n jobs are processed without interruption on some of the m equal machines belonging to the system; each machine can handle no more than one job at a time. Job j (j=1,...,n) becomes available for processing at time zero, requires an uninterrupted positive processing time p_j on a machine, and has a due date d_j by which it should ideally be finished. For a given processing order of the jobs, the earliest completion time C_j and the tardiness $T_j = \max\{C_j - d_j, 0\}$ of job j can readily be computed. The problem is to find a processing order of the jobs with minimum objective values. The objectives to be minimised are:

Average Tardiness: $$T_{av} = \frac{1}{n}\sum_{j=1}^{n} T_j$$

Weighted Tardiness: $$T_{wt} = \sum_{j=1}^{n} w_j T_j$$

Weighted Number of Tardy Jobs: $$N_{wt} = \sum_{j=1}^{n} w_j \delta(T_j), \text{ where } \delta(T_j) = 1 \text{ if } T_j > 0$$
$$\delta(T_j) = 0 \text{ otherwise}$$

These problems have received considerable attention by different researchers. For most of them, for many years their computational complexity remained as an open research topic until established as NP-Hard [16].

3 Conventional Approaches to Scheduling Problems

Dispatching heuristics assign a priority index to every job in a waiting queue. The one with the highest priority is selected to be processed next. There are different heuristics

[12] for the above mentioned problems whose principal property is not only the quality of the results but also to give an ordering of the jobs (schedule) close to the optimal sequence. The following dispatching rules and heuristics were selected to determine priorities, build schedules and contrast their outcomes with those obtained by the evolutionary algorithms.

LPT (Longest Processing Time first): The job with the longest processing time is selected first. The final scheduled jobs are ordered satisfying: $p_1 \geq p_2 \geq \ldots \geq p_n$.
WLPT (Weighted Longest Processing Time first): The job with the weighted longest processing time is selected first. The final scheduled jobs are ordered satisfying:
$(w_1/p_1) \leq (w_2/p_2) \leq \ldots \leq (w_n/p_n)$.

SPT (Shortest Processing Time first): The job with the shortest processing time is selected first. The final scheduled jobs are ordered satisfying: $p_1 \leq p_2 \leq \ldots \leq p_n$.
WSPT (Weighted Shortest Processing Time first): The job with the weighted shortest processing time is selected first. The final scheduled jobs are ordered satisfying:
$(w_1/p_1) \geq (w_2/p_2) \geq \ldots \geq (w_n/p_n)$.

EDD (Earliest Due Date first): The job with earliest due date is selected first. The final scheduled jobs are ordered satisfying: $d_1 \leq d_2 \leq \ldots \leq d_n$.

SLACK (Least slack): The job with smallest difference between due date and processing time is selected first. The final scheduled jobs are ordered satisfying:
$d_1\text{-}p_1 \leq d_2\text{-}p_2 \leq \ldots \leq d_n\text{-}p_n$.

Hodgson's Algorithm: This algorithm gives an optimal schedule for the unweighted number of tardy jobs objective and behaves well for some instances of average tardiness objective. The heuristic provides a schedule according to the following procedure,

Step 1: Order the activities using EDD heuristic.
Step 2: If there are no tardy jobs, stop; this is the optimal solution.
Step 3: Find the first tardy job, say k, in the sequence.
Step 4: Move the single job j $(1 \leq j \leq k)$ with the longest processing time to the end of the sequence.
Step 5: Check the completion times and return to step 2.

Rachamadugu and Morton Heuristic (R&M): This heuristic provides a schedule according to the following priority,

$$\pi_j = (w_j / p_j)\left[\exp\left\{-(S_j)^{+} / kp_{av}\right\}\right]$$

with $S_j = [d_j - (mp_j + Ch)]$ is the slack of job j at time Ch, where Ch is the total processing time of the jobs already scheduled, k is a parameter of the method (usually k =2.0) and p_{av} is the average processing time of jobs competing for top priority. In the *R&M* heuristic, jobs are scheduled one at a time. Every time a machine becomes free for each remaining job a new ranking index is computed. The job with highest ranking index is then selected to be processed next.

4 Multirecombination of Random and Seed Immigrants with the Stud

Multiple Crossovers per Couple (MCPC) [6,7] and Multiple Crossovers on Multiple Parents (MCMP) [8] are multirecombination methods, which improve EAs performance by reinforcing and balancing exploration and exploitation in the search process. In particular, MCMP is an extension of MCPC where the multiparent approach proposed by Eiben [3-5] is introduced. Results obtained in diverse single and multiobjective optimization problems indicated that the searching space is efficiently exploited by multiple applications of crossovers and efficiently explored by the greater number of samples provided by the multiple parents.

A further extension of MCMP is known as MCMP-SRI [13,14]. This approach considers the mating of an evolved individual (the stud) with random immigrants. The process for creating offspring is performed as follows. From the old population the stud is selected by means of proportional selection and inserted in the mating pool. The number of n_2 parents in the mating pool is completed with randomly created individuals (random immigrants). The stud mates every other parent, the couples undergo crossover operation and $2*(n_2-1)$ offspring are created. The best of these $2*(n_2-1)$ offspring is stored in a temporary children pool. The crossover operation is repeated n_1 times, for different cut points each time, until the children pool is completed. Finally, the best offspring created from n_2 parents and n_1 crossover is inserted in the new population.

As EAs are blind search methods our new variant (MCMP-SRSI) [15], proposes to insert problem-specific knowledge by recombining potential solutions (individuals of the evolving population) with seeds, which are solutions provided by other heuristics specifically designed to solve the scheduling problems under study. In MCMP-SRSI, the process for creating offspring is similar to that of MCMP-SRI, except that the mating pool contains also seed immigrants. In this way, the evolutionary algorithm incorporates problem-specific knowledge supplied by the specific heuristic. Figure 1 displays these processes.

We worked with different indirect representations: processor dispatching priorities and task priority list (both are indirect-decode representations) and another based on permutations.

The results discussed in next section correspond to EAs that worked on permutation-based representation using the PMX crossover operator because with this combination of representation and operator we obtained the best results.

5 Experimental Tests and Results

As it is not usual to find published benchmarks for the scheduling problems we worked on, we built our own test suite with data (p_j, d_j, w_j) based on 20 selected data corresponding to weighted tardiness problems of size 40 taken from the OR library [1,2].

These data were the input for dispatching rules, conventional heuristics and our proposed multirecombined studs, seeds and immigrants EAs, and SCPC.

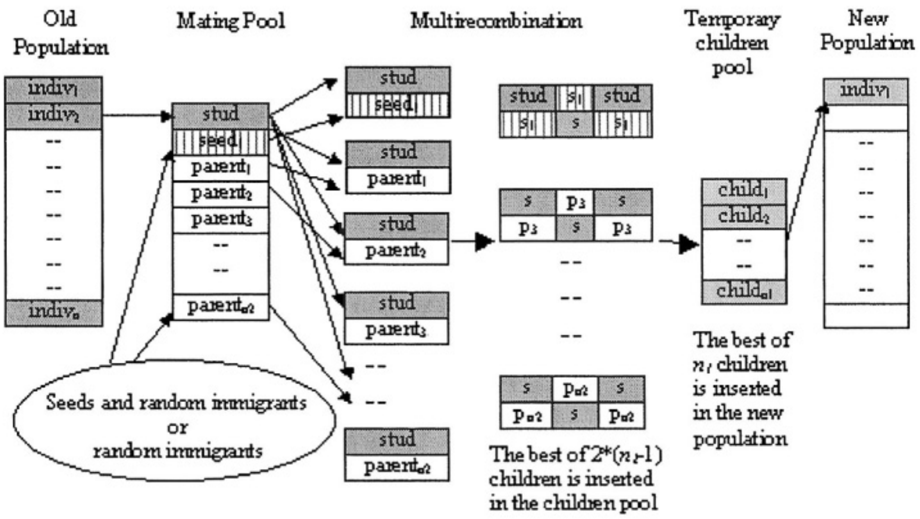

Old Population
Mating Pool
Multirecombination
Temporary children pool
New Population
indiv1
indiv2
indivn
stud
seed
parent1
parent2
parent3
parentn2
Seeds and random immigrants or random immigrants
The best of 2*(n1-1) children is inserted in the children pool
child1
child2
childn1
The best of n1 children is inserted in the new population

Fig. 1. The stud and (random immigrants/seeds and random immigrants) multirecombination processes

To evaluate the dispatching rules and the conventional heuristics we used PARSIFAL [12] a software package provided by Morton and Pentico, to solve different scheduling problems by means of different heuristics.

In this work, it is shown the improved performance of both multirecombinated EAs (MCMP-SRI and MCMP-SRSI) when compared with an EA without multirecombination, called SCPC (Simple Crossover Per Couple).

The initial phase of the experiments consisted in establishing the best results from dispatching rules and conventional heuristics to use them as upper bounds for the scheduling objectives. Also, the best parameter values for the EAs were empirically derived after performing a set of previous experiments. In all the experiments, we used population size 15 and we ran de EAs for 200 generations. The values of the remaining parameters are the following: crossover probability 0.65, $n_1 = 18$, $n_2 = 20$, seed number = 1 (only for MCMP-SRSI). For each problem and algorithm studied we performed 30 runs.

To compare the algorithms, the following relevant performance variables were chosen:

Ebest = ((best value - opt_val)/opt_val)*100

It is the percentile error of the best found individual when compared with the known or estimated (upper bound) optimum value opt_val. It gives a measure on how far the best individual is from that opt_val. When this value is negative, the op_val has been improved.

Mean Ebest (MEbe): It is the mean value of Ebest throughout all runs.

Mean Best (μBest): It is the mean objective value obtained from the best found individuals throughout all runs.

Mean Evals (MEvals): Is the mean number of evaluations necessary to obtain the best found individual throughout all runs. In our case, the results presented in this paper are expressed in units of thousands.

σBest: It is the standard deviation of the objective values corresponding to the best found individuals throughout all runs with respect to **μBest**.

(σ/μ) Best: This coefficient of variation it is calculated as the **σBest** and **μBest** ratio. It represents the deviation as a percentage of the value **μBest**. When this value is closer to zero higher is the robustness of the results obtained by the EA.

Several experiments were performed for 2 and 5 parallel equal machines scheduling systems for the objectives described before. The results obtained with the different multirecombinated EAs implementations performed well for both scheduling systems. In this paper, due to space constraints, we only present the best results obtained corresponding to the 5 parallel equal machines scheduling problem. Average values of 30 runs for some of the above performance variables, obtained under SCPC, MCMP-SRI and MCMP-SRSI approaches, are listed in the following tables. At the bottom of each table the mean values are showed. In all the tables the first column gives the numbers of the instances taken from the OR library. In the **μBest** column the values shown in italic bold case have improved the corresponding benchmark. In Tables 2 to 4, the Bench column displays the upper bound obtained from the respective heuristic in the third column.

In order to be able to compare the performance of an EA without multirecombination, with the multirecombinated EAs' performance, in Table 1 are presented the results found by SCPC, with 200 generations per run. There were also done experiments with 500, 1000, 2000, 3000, 4000 and 5000 generations. In view of the precision of the results obtained in those cases they were not much better, considering also that the average number of evaluations per run increases approximately twice as much.

For some instances, depending on the objective SCPC reached or improved the upper bound obtained by the conventional heuristics. Nevertheless, considering the precision of the results found by the multirecombinated EAs (MEbe column in Tables 2, 3 and 4), it is clear the advantage of using the proposed algorithms in the selected scheduling problems. It is important to note, that the cost (in number of evaluations) of these algorithms is much higher than the cost of SCPC. Thus, a decision should be made when facing this kind of scheduling problems; to obtain high precision results with a high cost or to obtain lower precision results at a moderate cost.

Tables 2, 3 and 4 show that the multirecombinated EAs also performed very well with respect to the upper bounds. For all the objective functions studied, the average values of MEbest for MCMP-SRSI are lower than the values obtained with MCMP-SRI with the additional advantage that MCMP-SRSI presented in all the cases lower average values for the mean number of evaluations (MEvals). For example, for the weighted tardiness objective (Table 3) the total average number of MCMP-SRI is in 1843.60 (in units of thousands), which is a value higher than the total average number of evaluations performed with MCMP-SRSI.

When considering the coefficient of variations of the best values found by the approaches presented (with and without multirecombination) we see that they group around the mean. Although not all the coefficients values are equal to 0.0 they are very close suggesting the algorithms' robustness with respect to the results that they found. In MCMP-SRI and MCMP-SRSI, their coefficients of variation are lower than the ones of SCPC, because of the reinforcement in selective pressure.

Table 1. SCPC EA performance for 40 jobs problems size

	Tavg				Twt				Nwt			
N°	MEbe	µBest	(σ/µ)	MEvals	MEbe	µBest	(σ/µ)	MEvals	MEbe	µBest	(σ/µ)	MEvals
1	17.69	38.84	0.08	4.97	25.54	4770.47	0.17	5.36	-28.53	***17.87***	0.19	3.59
6	-10.11	***126.74***	0.03	5.22	0.58	15991.50	0.13	5.64	-21.17	***42.57***	0.11	4.73
11	-1.64	***279.33***	0.03	5.57	14.28	34969.04	0.10	5.68	3.98	64.47	0.12	4.76
19	1.79	646.39	0.01	5.67	8.49	110655.16	0.03	5.72	-2.33	***135.77***	0.05	4.55
21	6.26	658.82	0.02	5.52	10.20	105568.13	0.03	5.74	1.02	164.67	0.02	1.85
26	104.42	18.40	0.28	5.04	29.22	3320.93	0.22	5.38	-30.48	***14.60***	0.26	3.61
31	11.76	146.41	0.06	5.56	4.65	21349.13	0.10	5.70	-18.56	***41.53***	0.09	4.17
36	-5.28	***279.43***	0.02	5.64	-6.16	***41475.13***	0.07	5.68	-13.00	***70.47***	0.08	4.74
41	1.29	502.42	0.02	5.62	2.89	88790.16	0.04	5.68	12.01	109.77	0.07	4.59
46	7.99	487.03	0.03	5.72	11.76	90636.60	0.04	5.76	-3.93	***183.50***	0.02	3.51
56	26.58	67.09	0.13	5.44	-45.51	***8172.90***	0.17	5.50	-48.10	***26.47***	0.17	4.48
61	-7.56	***268.09***	0.05	5.71	-2.87	***46430.20***	0.06	5.65	-20.07	***74.33***	0.09	5.18
66	2.58	556.01	0.03	5.66	7.31	102908.90	0.04	5.73	-14.95	***131.83***	0.06	4.64
71	4.93	583.41	0.02	5.79	8.49	123674.66	0.02	5.76	-7.94	***178.60***	0.03	3.31
86	13.30	215.26	0.05	5.70	7.55	44312.10	0.08	5.73	-25.39	***82.07***	0.11	4.74
91	-2.81	***474.30***	0.02	5.73	2.11	83732.73	0.05	5.71	-14.74	***103.17***	0.07	5.05
96	3.73	788.32	0.01	5.69	6.28	163667.67	0.02	5.70	-3.84	***184.63***	0.02	3.22
111	-0.37	***370.63***	0.03	5.72	-2.63	***67188.07***	0.05	5.68	-5.82	***89.47***	0.08	4.64
116	1.79	385.78	0.03	5.73	15.99	83393.90	0.04	5.69	-20.65	***117.43***	0.08	4.34
121	4.73	739.39	0.02	5.61	8.77	166420.50	0.02	5.70	0.48	180.87	0.03	4.13
AVG	**9.05**	**381.60**	**0.05**	**5.57**	**5.35**	**70371.39**	**0.07**	**5.66**	**-13.10**	**100.70**	**0.09**	**4.19**

Table 2. EAs performance for Average Tardiness 40 jobs problem size

			MCMP-SRSI					MCMP-SRI				
N°	Bench	Heuristic	MEbe	µBest	σBest	(σ/µ)	MEvals	MEbe	µBest	σBest	(σ/µ)	MEvals
1	**33**	R&M	-8.63	***30.15***	0.80	0.03	1117.24	-3.54	***31.83***	1.64	0.05	1267.86
6	**141**	SPT	-20.12	***112.63***	1.23	0.01	1608.54	-20.46	***112.15***	0.98	0.01	1563.66
11	**284**	SPT	-9.64	***256.63***	0.92	0.00	1441.60	-9.73	***256.37***	1.46	0.01	1641.86
19	**635**	SPT	-3.66	***611.73***	2.16	0.00	1609.22	-3.80	***610.87***	1.72	0.00	1889.04
21	**620**	SPT	-0.35	***617.84***	0.22	0.00	1355.24	-0.23	***618.56***	0.64	0.00	1814.24
26	**9**	R&M	-24.36	***6.81***	0.38	0.06	789.14	-19.28	***7.26***	0.55	0.08	1002.66
31	**131**	EDD	-12.10	***115.15***	0.52	0.00	1765.96	-9.70	***118.30***	2.71	0.02	1847.90
36	**295**	SPT	-15.27	***249.96***	3.48	0.01	1916.58	-16.08	***247.57***	2.36	0.01	1938.34
41	**496**	SPT	-6.65	***463.00***	1.12	0.00	1782.96	-6.64	***463.07***	1.03	0.00	1900.94
46	**451**	SPT	-0.76	***447.57***	0.19	0.00	1321.24	-0.55	***448.52***	0.71	0.00	1851.64
56	**53**	EDD	-22.03	***41.32***	0.54	0.01	1358.98	-14.24	***45.45***	1.12	0.02	1759.16
61	**290**	R&M	-22.90	***223.58***	1.82	0.01	1892.78	-23.01	***223.26***	2.22	0.01	1913.86
66	**542**	SPT	-8.19	***497.59***	1.26	0.00	1806.08	-7.94	***498.96***	1.48	0.00	1876.46
71	**556**	SPT	-2.41	***542.58***	0.48	0.00	1752.36	-2.28	***543.30***	0.84	0.00	1866.60
86	**190**	HODG.	-13.99	***163.42***	1.95	0.01	1850.28	-14.12	***163.18***	1.89	0.01	1911.82
91	**488**	SPT	-11.69	***430.96***	1.74	0.00	1850.62	-11.71	***430.84***	1.60	0.00	1900.94
96	**760**	SPT	-1.36	***749.68***	0.57	0.00	1664.64	-1.31	***750.08***	0.59	0.00	1877.48
111	**372**	R&M	-17.17	***308.15***	2.44	0.01	1953.98	-16.97	***308.86***	2.79	0.01	1952.28
116	**379**	SPT	-9.96	***341.26***	1.96	0.01	1864.90	-10.29	***340.01***	1.43	0.00	1947.52
121	**706**	SPT	-2.32	***689.61***	0.48	0.00	1599.02	-2.22	***690.35***	0.80	0.00	1818.32
		AVG	**-10.68**	**344.98**	**1.21**	**0.01**	**1615.07**	**-9.71**	**345.44**	**1.43**	**0.01**	**1777.13**

Table 3. EAs performance for Weighted Tardiness 40 jobs problem size

			MCMP-SRSI					MCMP-SRI				
N°	Bench	Heuristic	MEbe	µBest	σBest	(σ/µ)	MEvals	MEbe	µBest	σBest	(σ/µ)	MEvals
1	**3800**	R&M	-34.43	***2491.57***	214.36	0.09	1226.72	-34.70	***2481.47***	197.10	0.08	1626.90
6	**15900**	WSPT	-32.14	***10789.37***	215.57	0.02	1725.50	-32.15	***10788.60***	161.87	0.02	1739.10
11	**30600**	WSPT	-16.49	***25554.53***	266.87	0.01	1612.62	-16.58	***25527.37***	261.54	0.01	1775.14
19	**102000**	WSPT	-4.67	***97232.16***	307.03	0.00	1734.68	-4.57	***97342.10***	306.40	0.00	1891.08
21	**95800**	R&M	-0.15	***95660.37***	29.11	0.00	1657.16	0.06	95860.57	108.56	0.00	1953.98
26	**2570**	R&M	-63.03	***950.03***	307.34	0.32	1156.00	-61.56	***988.00***	314.24	0.32	1370.54
31	**20400**	R&M	-42.09	***11814.07***	277.28	0.02	1761.20	-40.64	***12109.97***	638.43	0.05	1859.12
36	**44200**	WSPT	-31.84	***30125.50***	891.75	0.03	1885.30	-32.82	***29695.10***	674.90	0.02	1895.84
41	**86300**	WSPT	-11.96	***75974.87***	487.31	0.01	1785.00	-11.69	***76208.43***	442.93	0.01	1856.06
46	**81100**	WSPT	-0.54	***80661.13***	40.31	0.00	1693.54	-0.32	***80844.07***	95.05	0.00	1941.06
56	**15000**	EDD	-72.41	***4137.97***	342.04	0.08	1580.66	-71.98	***4203.13***	363.34	0.09	1673.14
61	**47800**	R&M	-28.41	***34219.77***	527.87	0.02	1915.22	-28.91	***33979.10***	438.29	0.01	1904.00
66	**95900**	WSPT	-9.58	***86711.03***	255.22	0.00	1801.66	-9.36	***86920.16***	280.85	0.00	1907.40
71	**114000**	R&M	-1.23	***112598.73***	69.78	0.00	1767.32	-1.04	***112812.57***	133.91	0.00	1921.68
86	**41200**	HODG.	-40.20	***24636.10***	750.69	0.03	1866.94	-40.70	***24429.60***	618.85	0.03	1893.46
91	**82000**	R&M	-16.41	***68545.13***	327.05	0.00	1897.54	-16.22	***68700.30***	296.31	0.00	1953.98
96	**154000**	R&M	-1.46	***151753.27***	111.06	0.00	1773.78	-1.38	***151872.20***	193.80	0.00	1953.30
111	**69000**	R&M	-25.79	***51201.77***	512.17	0.01	1949.90	-25.90	***51127.96***	478.44	0.01	1916.92
116	**71900**	R&M	-3.92	***69082.50***	408.93	0.01	1867.96	-4.20	***68881.16***	380.58	0.01	1946.16
121	**153000**	R&M	-1.72	***150370.44***	179.70	0.00	1719.04	-1.56	***150612.70***	235.67	0.00	1893.12
		AVG	**-21.92**	**59225.52**	**326.07**	**0.03**	**1718.89**	**-21.81**	**59269.23**	**331.05**	**0.03**	**1843.60**

Table 4. EAs performance for Weighted Number of Tardy Jobs 40 jobs problem size

			MCMP-SRSI					MCMP-SRI				
N°	Bench	Heuristic	MEbe	µBest	σBest	(σ/µ)	MEvals	MEbe	µBest	σBest	(σ/µ)	MEvals
1	**25**	WSPT	-57.60	***10.60***	1.07	0.10	719.44	-56.93	***10.77***	1.01	0.09	648.04
6	**54**	WSPT	-56.67	***23.40***	2.30	0.10	762.28	-54.14	***24.77***	2.36	0.10	1031.22
11	**62**	WSPT	-32.80	***41.67***	2.22	0.05	714.34	-27.90	***44.70***	2.68	0.06	1107.04
19	**139**	WSPT	-15.30	***117.73***	2.32	0.02	687.82	-14.34	***119.07***	2.52	0.02	925.48
21	**163**	SPT	-0.61	***162.00***	0.00	0.00	21.08	-0.61	***162.00***	0.00	0.00	42.84
26	**21**	EDD	-81.90	***3.80***	1.00	0.26	698.36	-79.05	***4.40***	1.00	0.23	718.76
31	**51**	WSPT	-49.15	***25.93***	2.42	0.09	819.74	-48.76	***26.13***	2.40	0.09	1144.44
36	**81**	SPT	-30.25	***56.50***	4.07	0.07	983.28	-38.02	***50.20***	2.94	0.06	1245.42
41	**98**	WSPT	-17.41	***80.93***	3.33	0.04	821.78	-15.48	***82.83***	4.33	0.05	870.40
46	**191**	HODG.	-6.28	***179.00***	0.00	0.00	112.20	-6.28	***179.00***	0.00	0.00	93.16
56	**51**	WSPT	-75.36	***12.57***	2.70	0.21	1010.48	-75.95	***12.27***	2.26	0.18	865.98
61	**93**	WSPT	-51.47	***45.13***	3.15	0.07	1228.08	-50.50	***46.03***	2.41	0.05	1212.78
66	**155**	WSPT	-32.80	***104.17***	3.46	0.03	888.42	-31.59	***106.03***	3.40	0.03	1094.12
71	**194**	SPT	-13.95	***166.93***	1.64	0.01	573.58	-13.90	***167.03***	2.08	0.01	635.46
86	**110**	WSPT	-58.03	***46.17***	5.42	0.12	1351.16	-55.88	***48.53***	5.07	0.10	1456.22
91	**121**	WSPT	-35.70	***77.80***	3.60	0.05	880.60	-35.87	***77.60***	2.43	0.03	895.22
96	**192**	WSPT	-7.81	***177.00***	0.00	0.00	198.56	-7.81	***177.00***	0.00	0.00	183.26
111	**95**	WSPT	-21.23	***74.83***	2.53	0.03	681.70	-24.35	***71.87***	1.96	0.03	834.36
116	**148**	SPT	-38.29	***91.33***	4.03	0.04	946.22	-39.05	***90.20***	2.78	0.03	853.74
121	**180**	HODG.	-5.98	***169.23***	0.77	0.00	571.88	-6.07	***169.07***	0.25	0.00	387.26
		AVG	**-34.43**	**83.34**	**2.30**	**0.06**	**733.55**	**-34.12**	**83.48**	**2.09**	**0.06**	**812.26**

In Figure 2, are shown the **μBest** values corresponding to Table 3. Although the differences among the multirecombinated methods are small and therefore, difficult to appreciate them graphically, for this objective, SCPC reached the lowest number of instances' benchmarks improved.

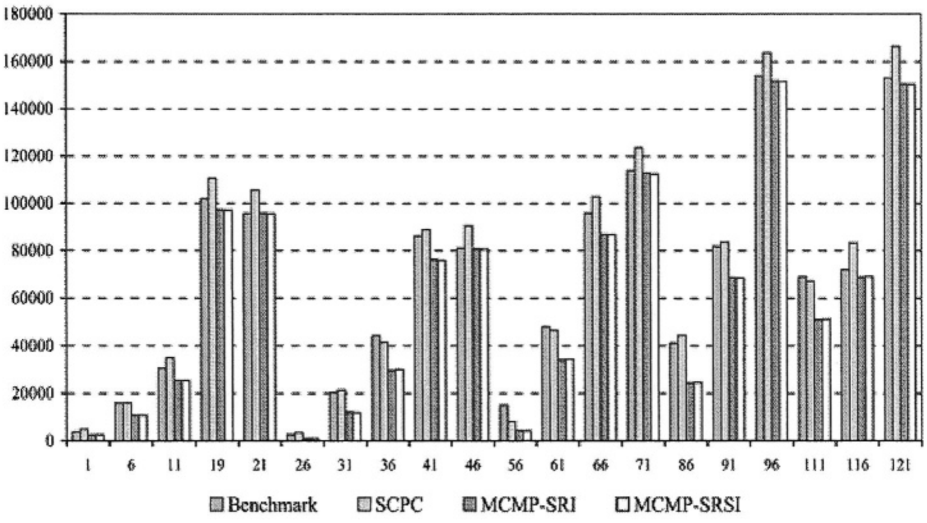


Fig. 2. Weighted Tardiness MBest values found by the EAs

6 Conclusions

Multirecombinated evolutionary algorithms have been successfully used to solve scheduling problems. In particular MCMP-SRI and MCMP-SRSI, the approaches considered here have demonstrated their ability on unrestricted parallel machine due date related scheduling problems, by improving the results found by an EA without multirecombination and the upper bounds calculated with different conventional heuristics using PARSIFAL [12] for various problem data taken from the OR-Library.

It is known that in some cases seeding individuals in the population is equivalent of running for a few more generation the EA without seeding individuals. However, in our studies we have concluded that this is true for small and medium scheduling problem sizes, and also depends on the hardness of problem instance and the type of EA used. When comparing MCMP-SRSI with MCMP-SRI, the former finds good results with a lower cost (number of evaluations), due to the knowledge of the problem used to guide the search to promising areas of solutions. In this way using seeds in the population instead of running the EA for more generations has the advantage of speeding the convergence of the EA towards good solutions at a lower cost.

Future work will be devoted to solve due date related problems in parallel machine scheduling systems for larger number of jobs and to compare the performance of the different EAs implemented with others population-based stochastic search heuristics.

Acknowledgements

We acknowledge the Universidad Nacional de San Luis and the ANPCYT from which we receive continuous support.

References

1. J.E. Beasley "Weighted Tardiness", OR Library, http://mscmga.ms.ic.ac.uk/
2. H.A.J. Crauwels, C.N. Potts and L.N. Van Wassenhove, "Local Search Heuristics for the Single Machine Total Weighted Tardiness Scheduling Problem", Informs Journal on Computing 10, pp. 341-350, 1998.
3. Eiben A.E., Raué P.E., and Ruttkay Z., "Genetic Algorithms with Multi-parent Recombination", Proceedings of the 3rd Conference on Parallel Problem Solving from Nature, number 866 in LNCS, pp. 78-87, Springer-Verlag, 1994.
4. Eiben A.E., Van Kemenade C.H.M., and Kok J.N., "Orgy in the Computer: Multiparent Reproduction in Genetic Algorithms", Proceedings of the 3rd European Conference on Artificial Life, number 929 in LNAI, pp. 934-945, Springer-Verlag, 1995.
5. Eiben A.E. and. Bäck T., "An Empirical Investigation of Multi-parent Recombination Operators in Evolution Strategies", Evolutionary Computation, 5(3):347-365, 1997.
6. Esquivel S., Leiva A., Gallard R., "Multiple Crossover per Couple in Genetic Algorithms", Evolutionary Computation (ICEC'97), IEEE Publishing Co, pp. 103-106, ISBN 0-7803-3949-5, Indianapolis, USA, April 1997.
7. Esquivel S., Leiva A., Gallard R., "Couple Fitness Based Selection with Multiple Crossover Per Couple in Genetic Algorithms", Proceedings del International Symposium on Engineering of Intelligent Systems, University of La Laguna, España, Tenerife, Vol. 1, pp 235-241, ISBN 3-906454-12-6, February 1998.
8. Esquivel S., Leiva H., Gallard R., "Multiple Crossovers between Multiple Parents to Improve Search in Evolutionary Algorithms", Evolutionary Computation, IEEE Publishing Co, Washington DC, pp. 1589-1594, 1999.
9. Esquivel S., Ferrero S., Gallard R., Salto C., Alfonso H. and Schütz M., "Enhanced Evolutionary Algorithms for Single and Multiobjective Optimization in the Job Shop Scheduling Problem", Knowledge-Based Systems 15 (2002), pp. 13-25.
10. Feltl H. and Raidl G., "An Improved Hybrid Genetic Algorithm for the Generalized Assignment Problem", In Proceedings of the 2004 ACM symposium on Applied computing, Nicosia, Cyprus, ECO, pp. 990–995, ISBN:1-58113-812-1, 2004.
11. Min Liu and Cheng Wu, "Scheduling Algorithm based on Evolutionary Computing in Identical Parallel Machine Production Line", Robotics and Computer-Integrated Manufacturing 19 (2003), pp. 401-407.
12. Morton T., Pentico D., "Heuristic Scheduling Systems", Wiley series in Engineering and technology management, John Wiley and Sons, INC, 1993.
13. Pandolfi D., Vilanova G., De San Pedro M., Villagra A., Gallard R., "Multirecombining Studs and Immigrants in Evolutionary Algorithm to face Earliness-Tardiness Scheduling Problems", In Proceedings of the International Conference in Soft Computing, pp. 138, University of Paisley, Scotland, U.K., June 2001.
14. Pandolfi D., De San Pedro M., Villagra A, Vilanova G., Gallard R., "Studs Mating Immigrants in Evolutionary Algorithm to Solve the Earliness-Tardiness Scheduling Problem", In Cybernetics and Systems of Taylor and Francis Journal, Vol.33 Nro. 4, pp. 391-400 (U.K.), June 2002.

15. Pandolfi D., De San Pedro M., Villagra A., Vilanova G., Gallard R., "Multirecombining Random and Seed Immigrants in Evolutionary Algorithms to Solve W-T Scheduling Problems", In proceedings of CSITeA02, pp 133-138, Iguazu Falls, Brazil, June 2002.
16. Pinedo M., "Scheduling: Theory, Algorithms and System", Prentice Hall, First edition, 1995.

An Investigation on Genetic Algorithms for Generic STRIPS Planning

Marcos Castilho[1], Luis Allan Kunzle[2], Edson Lecheta[1], Viviane Palodeto[1], and Fabiano Silva[2]

[1] Departamento de Informatica, Federal University of Parana, Curitiba, Brazil
{marcos, lecheta, vp99}@inf.ufpr.br
http: //www. inf.ufpr. br/~marcos
[2] CPGEI/CEFET-PR, Curitiba, Brazil
{kunzle, fabiano}@cpgei.cefetpr.br
http://www.cpgei.cefetpr.br/~fabiano

Abstract. We investigate the technique of genetic algorithms to solve the class of *STRIPS* planning problems in Artificial Intelligence. We define a genetic algorithm called *AgPlan* and we compare it with some recent planners.

Keywords: Planning, Evolutionary Computation.

1 Introduction

Given a formal description of the behaviour of a set of actions, to find a sequence of actions which lead from a known state of the world to some desired one is defined as the *Planning Problem in Artificial Intelligence.* When the formal language used to describe actions is *STRIPS* [1] we call it *STRIPS Planning.* This restricted class of problems was proved to be PSPACE-complete [2].

For more than twenty years the classical AI search techniques were not sufficient to find solutions even for simple problems as the Sussman anomaly, a simple problem with three blocks in the well known blocks world domain. One main problem is that the search space is really huge. Moreover, there were no good heuristic functions to guide the process. The proposals failed reaching very easily in local extremes.

Two recent approaches changed that picture. One was *SATPLAN* whose main feature was to relate planning with satisfiability [3]. The other one was *GRAPHPLAN*[4]. It is based on a structure called the *plan graph* which successfully reduced the huge search space to a smaller part of it where a solution can more easily be found.

Since then research in the planning field has known an extraordinary growth. Several other areas of research contributed with important algorithms and tools. We can cite integer programming [5], constraint satisfaction [6], Petri nets [7], among others. Also, ancient techniques were reintroduced, e.g. the system *R,* a new implementation of the original *STRIPS,* has manage to solve really huge instances of difficult problems as the blocks world [8]. Of particular interest to us, very good heuristic functions were finally found and the heuristic search planners *HSP* [9] and *FF* [10] are among the best planners known today. Although some of these algorithms are very fast in general and are capable of dealing with large instances of problems, there are still open research to be done.

Genetic algorithms [11] (GA's) are usually considered as a possible technique to apply when the search space is huge. Surprisingly, little has been done to improve genetic techniques in domain independent planning. Main contributions seems to be related with robotics, but in terms of genetic programming [12, 13]. However, in the planning research it is usually considered that planners must be general purpose [14]. In this sense, as far as we could investigate, there is no generic planner based on genetic algorithms. In this paper we show the results of our research in this direction.

The paper is organised as follows. Section 2 contains some basic points about planning. In section 3 we introduce genetic algorithms in the context of planning and we present the details of our genetic algorithm. In section 4 we show how it works when applied to few well known domains used in the planning competitions.

2 The Planning Problem

The planning problem in Artificial Intelligence is, basically, to determine a sequence of actions which, when applied into some known initial state, leads to some desired final state. This problem is considered to be domain independent. The planner must be capable to read a planning problem consisting of a description of the behaviour of the actions and a description of the initial and the final situation. It must return a sequence of actions that, when performed in the given order to the initial situation, transforms it into the final situation. In our case, the planning problems are described in *PDDL* [15] that is considered today the common generic language to the planning community.

Any sequence of actions that do so is called a *plan.* Unfortunately, there is no fixed size for the plan. In most situations, it is supposed that the planner must return the optimal plan. On the other hand, as it is difficult, just to find some plan is considered to be enough. Usually, the planner must run with limited memory and time. In this paper we consider that any plan which solve the problem is a solution. We do not consider questions related to the quality of the plan or the number of actions used. In other terms, we do not consider the planning problem as an optimisation problem.

As we have said in the introduction there are several reasons for which we think genetic algorithms apply well in planning problems, although it is not a trivial task to elaborate the encodings of the GA. The next section describes this task in detail.

3 Preliminary Discussion on Genetic Planners

In this section we recall the main points. A more complete description may be found in [16]. In the next section we present alternative definitions.

3.1 Chromosome Encoding

The first basic point is how to encode a chromosome. Chromosomes are candidates of solution to the problem being solved, in our case, they are possible plans. In this way each gene will represent a single action of the plan. Consequently, the whole chromosome is the full plan, i.e., a complete solution. An action is represented by an integer (from 1 to the possible number of actions). Here comes the first difficult decision: plans are sequence of actions with no pre-specified length and in general chromosomes have fixed size.

Our first decision was to simulate variable length size using fixed-length chromosome with the inclusion of a new action in the list of possible actions ("NOP" – no operation). This is a null action that can be applied in any point of a plan and has no effect to the current state of the world. With the use of this extra action a given plan can be extended as wanted, without loosing its original effect. Variable-length plans were then handled by fixed-length chromosomes, by setting up an upper bound length for plans and filling them with NOP's. This decision was well suited for our experiments, since we were using a standard package called *pgapack*[1].

But we found in practice that we spent most of the time taking care of the simulation process and the planner was too slow. The final decision was to define a variable-length chromosome. The impact of this decision was that we decided to abandon that package and to develop all the code by ourselves.

3.2 The Fitness Function

The fitness function measures the quality of a chromosome, it must evaluate the individuals which are in fact plans with the high possible value. The question is: how do we know that a sequence of actions is a plan? The answer is that the chromosome must be validated starting with the action represented by its first gene been applied to the initial state and sequentially all other actions represented by the others genes until the final one obtains the goal state of the problem.

Suppose a chromosome with size n and genes $g_1, g_2, \ldots, g_n$. It may happen that g_1 is not applicable to the initial state, but g_2 is. It is possible that g_3 is not applicable in the state resulting for the application of g_2, but g_4 is. One possible decision is to not allow this kind of situation. We have however take another way.

Our decision was to define an auxiliary binary vector parallel to the chromosome to give information about executability of the actions represented by

[1] http://www-fp.mcs.anl.gov/CCST/research/reports_pre1998/ /comp_bio/stalk/pgapack.html

the genes with relation with the initial state. In this way, let's consider the first non-zero element in this binary auxiliary vector. Let's say that it is in position i. This means that all actions associated with the genes in positions 1 to $i-1$ are not applicable to the initial state, but that on the ith position is. Let's now say that the next non-zero element in the auxiliary vector is in position j. This means that all genes representing actions in the chromosome starting in position $i+1$ until position $j-1$ are not applicable to the state resulting from the application of the previous valid action but that at the associated j position is.

But this is not enough, we need to measure how far a sequence of actions is from been a solution, if it is not one. Giving that we have two vectors, one containing a candidate to a solution and the other containing information about the executability of actions in the first vector, our fitness function is a combination of informations extracted from both vectors.

Another point of consideration is that it may happen that a chromosome is not a solution, but it is a good sequence of actions that lead us closely enough of a solution. They may be considered good beginnings of plans. It may happen, on the other hand, that it a very good sequence of actions which obtains the final state, but not from the initial state. They may be considered good final of plans. We then let for the genetic operator of crossover to put the good parts together forming a complete plan.

Now, our fitness function will be a combination of the following information:

1. the number of executable actions in the progressive direction, that is, the number of non-zero elements on the vector of executability when the validation is analysed from the initial state;
2. the number of executable actions in the regressive direction, that is, the number of non-zero elements on the vector of executability when the validation is analysed from the final state;
3. the number of propositions of the goal state reached when the associated plan is applicable from the initial state;
4. the number of propositions of the initial state reached when the associated plan is applicable from the final state;

Actually we have tried with several other informations, like, e.g.:

1. the number of consecutive executable actions;
2. the number of valid actions;
3. the position of the first executable action;
4. the position of the last executable action;
5. the number of actions of the largest feasible subsequence (fragment of the plan) from the beginning of it;
6. the number of applicable actions of the largest partially feasible subsequence, from the beginning of it. This factor is different from the previous because it allows non-applicable actions being present in an feasible subsequence;

The idea was to obtain high quality index for a candidate solution based on its number of feasible actions. The larger the number of feasible actions an individual, and the more organised they are, the higher the quality it would have. It is in fact difficult to establish a good fitness function. If we try to maximise executability the resulting chromosomes have lots of useless sequences of actions, as unloading the truck after having loaded it. If we try to give high values to individual with leads to a great number of goals, then we get no solution.

3.3 The Genetic Operators

We define three operators: mutation, crossover and a new one called *compression.*

The classical mutation operator was used to assure an even exploration of the search space, aiming to avoid the algorithm gets stuck in local extremes. This operator randomly chooses a gene and changes its value to a random integer ranging from 0 to the size of the list of possible actions. For every gene of the chromosome, this operator is applied with a given small probability. This is shown in figure 1.

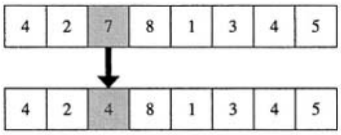


Fig. 1. Example of mutation

For the crossover operator, it is in general implemented in such a way that some point is randomly chosen in the chromosome and this point is applied to both parents. But here, for the reason that we had defined variable-length size chromosomes, we decided to implement a different version. In our work we randomly choose two points referring respectively to the recombination point for the first and second individual. This is shown in figure 2.

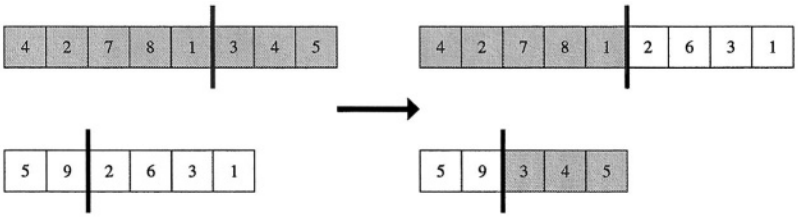


Fig. 2. Example of our crossover

Due to the way we had chosen to construct the chromosome, allowing for them to have non-executable sequences of actions (for this reason we have to work with a parallel binary vector for executability), it was necessary to define a new operator called *compression.* The reason is that in some moments, our chromosomes are too big. This operator looks for null elements in the binary vector of executability which, when found, are removed from both vectors, the one defining the chromosome and the one of binary executability of actions. This is shown in figure 3.

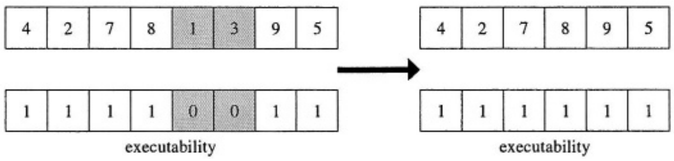


Fig. 3. The compression operator

3.4 The Running Parameters

In this section we describe the way we initialise the initial population, the method of replacement of the population and the rates of application of the genetic operators.

There are two possible ways to generate the initial population. The first is to let it be chosen randomly. The other way is to have some control on the process. In all the experiments we have done with an randomly initialised initial population we get very disappointing results for every possible combination of the genetic parameters. To control the generation of the initial population we used the plan graph, in its relaxed and full versions. The idea is to choose actions appearing in the initial layers as the first genes and actions of the last layers appearing in the last part of the chromosomes. As we will show in the next section this has a great impact in our approach.

The algorithm is as follows: construct the plan graph and for each layer randomly choose a number of actions to be randomly chosen in that layer to compose the chromosome. Do it for all layers. This will give us individuals with different size.

The selection method has a great impact in the convergence of the GA. In this work we decided to replace the entire population.

The genetic operators were applied with the following probabilities: 1% for mutation, 80% for crossover, which are within the usual range for GA's, and 30% for compression. These values were obtained after a number of experiments.

For the fitness function we get our best results with values in the range 0.4 to 0.45 for the two parameters controlling the number of goals (backwards and forwards) and values in the range 0.05 to 0.1 to the variables controlling the executability of actions). In the majority of cases we evaluate both forwards and backwards.

It must be said however that the most important running parameter is the plan graph based generation of the initial population. The GA finishes when we have found a solution or a previously specified number of maximal generation was found.

In the next section we show some experiments running the GA with different running parameters for the size of initial population and number of generations.

4 Experiments

We have done several experiments with *AgPlan* and we compare it with some of the best known planners, all of them based on the plan graph construction:

– *IPP:* implements the *GRAPHPLAN* algorithm [17];
– *BLACKBOX:* replaces the second phase of *GRAPHPLAN* by a translation to a SAT instance and then solve it using walksat[18];
– *FF:* it implements a heuristic search planner and is considered the faster of the actual planners. The heuristic function is based on the relaxed plan graph [10].

We choose three classical scenarios used in the planning competitions: logistics, gripper [19] and blocks-world [20]. For lack of space we will show only part of our experiments, giving two instances of each problem, including some easy and some difficult ones.

AgPlan implements the GA described in section 3 in this paper. *AgPlan*-r implements the GA with randomly initialised initial population. *AgPlan-m* implements the generation of the initial population using the complete plan graph, i.e., considering the mutex relation.

The implementation was realized in C and C++ languages for the GNU - Linux environment. The tests were run in a dual Xeon with 2.6GHz with 4GB of RAM by December 2003 [21] as part of a graduate course on Planning in AI.

4.1 Results for Gripper

Figure 4 we show the results for the gripper problem 22 balls (left table) and 42 balls (right table). They are considered very difficult problems. In both cases we have used 30 as the maximum number of generations. We have used population of size 700 for the problem with 22 balls and 3600 for other. We analyse the size of the resulting plan and the time to obtain it.

Only *FF* and *AgPlan* have managed to solve these two huge instances of problems. It is true that *FF* is faster, but it may be considered a good result for *AgPlan.* Moreover, we can see the importance of using the plan graph to construct the initial population, *AgPlan*-r had no good results. The initialisation considering mutex seems to make the algorithm getting slow.

The plans are longer than those of *FF,* but a careful analysis shows that most of the actions are of the type “load” followed by “unload”. This can be solved as a post-processing step, as it is done for some planners, e.g., the system *R* [8].

4.2 Results for Blocks World

Figure 5 shows our experiments with the classical Sussman anomaly (left table) and another using 8 blocks to be put in the inverse position (right table). The population had up to 9000 to solve the latter. They are presented (in order) below.

AgPlan seems to apply well in this domain. Even the version with randomly initialised initial population run faster then *IPP* and *BLACKBOX.* It has found

Planner	plan size	time (ms)
IPP	–	–
BLACKBOX	–	–
FF	65	0.00
AgPlan	81	0.72
AgPlan-r	–	–
AgPlan-m	85	0.57

Planner	plan size	time (ms)
IPP	–	–
BLACKBOX	–	–
FF	125	2.00
AgPlan	163	5.69
AgPlan-r	–	–
AgPlan-m	187	12.07

Fig. 4. Experiments with gripper

Planner	plan size	time (ms)
IPP	3	0.00
BLACKBOX	3	0.07
FF	3	0.00
AgPlan	3	0.00
AgPlan-r	3	0.00
AgPlan-m	3	0.00

Planner	plan size	time (ms)
IPP	11	20.00
BLACKBOX	17	88.81
FF	11	0.00
AgPlan	10	0.31
AgPlan-r	14	8.22
AgPlan-m	10	0.38

Fig. 5. Experiments with blocks-world

a shorter plan than *FF*. Here the initial population seems to have no other importance except that for the random case which are slower and have bigger plans. The results are satisfactory.

4.3 Results for Logistics

The only planners that managed to solve logistics problems were *FF* and *IPP*. This latter solves only two of them. The reason seems to be the number of layers in the relaxed plan graph, which is very small due to the high rates of parallelism of its actions.

5 Conclusion

In this paper, a methodology for dealing with planning problems using Genetic Algorithms was presented. The proposed GA has special features specially devised for such kind of problem, including a variable-length chromosome for variable-size plans, and a special genetic operator, namely, compression. We have also implemented a special version of crossover with two points.

Our planner is general purpose and works only with typed versions of the scenarios. Previous experiments shows that the non-typed versions give us very poor results. One possible way to overcome that is to implement automatic type verification. This problem was investigated by [22] and [16].

This methodology is still under development, but the results are encouraging. Although many experiments are yet to be done, we believe that it will be particularly interesting for problems that cannot be addressed with current traditional methods. In particular, based on hundreds of experiments, we think that the generation of the initial population based on the plan graph construction is the key for the success of our GA.

Future work includes some experiments with different genetic parameters in order to improve the algorithm for harder problems, specially those involving domains with huge number of parallel actions, as logistics. This may include the use of local search or island parallel genetic algorithms.

Acknowledgements. This work has its origin in a graduate course on genetic algorithms on the CPGEI-CEFET/PR and on a graduate course on planning in the Department of Informatics/UFPR. We are very thankful to all the students that contributed to our results, in particular Daniel Benke, Vinicius Cubas and Tiago Lima. We are specially thankful to prof. Heitor Lopes, with whom we have several and useful discussions in early work on this subject. We also thanks CAPES for the PhD grant for Fabiano and SESU for the PET grant for Viviane.

References

1. Fikes, R., Nilsson, N.: STRIPS: A new approach to the application of theorem proving to problem solving. Journal of Artificial Intelligence **2** (1971)
2. Bylander, T.: The computational complexity of propositional STRIPS planning. Artificial Intelligence **69** (1994) 165–205
3. Kautz, H., Selman, B.: Planning as satisfiability. In: Proc. 10th Eur. Conf. AI, Vienna, Austria, Wiley (1992) 359–363
4. Blum, A., Furst, M.: Fast planning through planning graph analysis. In: Proceedings of IJCAI-95, Montreal (1995) 1636–1642
5. Vossen, T., Ball, M., Lotem, A., Nau, D.: On the use of integer programming models in ai planning. In: Proceedings of the IJCAI 99. (1999) 304–309
6. Beek, P., Chen, X.: CPlan: A constraint programming approach to planning. In: Proc. of the Sixteenth National Conference on AI (AAAI-99). (1999) 585–590
7. Silva, F., Castilho, M., Künzle, L.: Petriplan: a new algorithm for plan generation (preliminary report). In: Proc. of IBERAMIA/SBIA-2000, Springer-Verlag (2000) 86–95
8. Lin, F.: A planner called r. In: AIPS-2000 Planning Competition. (2000)
9. Bonet, B., Geffner, H.: HSP: Planning as heuristic search. In: Entry at the AIPS-98 Planning Competition, Pittsburgh (1998)
10. Hoffmann, J., Nebel, B.: The FF planning system: Fast plan generation through heuristic search. In: AIPS Planning Competition. (2000)
11. Goldberg, D.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley (1989)
12. Westerberg, C., Levine, J.: GenPlan: Combining genetic programming and planning. In: Proceedings for the UK Planning and Scheduling Special Interest Group, Milton Keynes UK (2000)
13. Muslea, I.: SINERGY: A linear planner based on genetic programming. In: Proc. European Conference on Planning (ECP-97), Toulouse, France (1997) 314–326
14. Geffner, H.: Perspectives on artificial intelligence planning. In: Proc. 18th Nat. (US) Conference on AI. (2002) 1013–1023
15. McDermott, D.: PDDL - The Planning Domain Definition Language. (1998) http://ftp.cs.yale.edu/pub/mcdermott.
16. Lecheta, E.M.: Algoritmos genticos para planejamento em inteligncia artificial. Master's thesis, Dinf-UFPR (2004)

17. Koehler, J.: IPP home page (1997)
18. Kautz, H., Selman, B.: Unifying sat-based and graph-based planning. In: Proceedings of IJCAI, Stockholm, Sweden, Morgan Kaufmann (1999) 318–325
19. McDermott, D.: AIPS-98 Planning Competition Results. (1998) http://ftp.cs.yale.edu/pub/mcdermott/aipscomp-results.html.
20. : Aips-2000 planning competition (2000)
21. Benke, D., Brand, V., Lima, S.: Uma avaliao de desempenho de planejadores. Technical report, Departamento de Informtica - UFPR, Curitiba, Brasil (2003)
22. Long, D., Fox, M.: Automatic synthesis and use of generic types in planning. In: Artificial Intelligence Planning Systems. (2000) 196–205

Improving Numerical Reasoning Capabilities of Inductive Logic Programming Systems

Alexessander Alves, Rui Camacho, and Eugenio Oliveira

LIACC, Rua do Campo Alegre, 823, 4150 Porto, Portugal
FEUP, Rua Dr Roberto Frias, 4200-465 Porto, Portugal
alves@ieee.org, {rcamacho, eco}@fe.up.pt
tel: +351 22 508 184 fax: +351 22 508 1443

Abstract. Inductive Logic Programming (ILP) systems have been largely applied to classification problems with a considerable success. The use of ILP systems in problems requiring numerical reasoning capabilities has been far less successful. Current systems have very limited numerical reasoning capabilities, which limits the range of domains where the ILP paradigm may be applied.

This paper proposes improvements in numerical reasoning capabilities of ILP systems. It proposes the use of statistical-based techniques like Model Validation and Model Selection to improve noise handling and it introduces a new search stopping criterium based on the PAC method to evaluate learning performance.

We have found these extensions essential to improve on results over statistical-based algorithms for time series forecasting used in the empirical evaluation study.

1 Introduction

Inductive Logic Programming (ILP) [1] has achieved considerable success in domains like biochemistry [2], and language processing [3]. The success of those applications are mainly due to the intelligibility of the models induced. Those models are expressed in the powerful language of first order clausal logic. In the domains just mentioned, the background knowledge is mainly of a relational nature. Theoretically there is no impediment of using whatever knowledge is useful for the induction of a theory. For some applications it would be quite useful to include as background knowledge methods and algorithms of a numerical nature. Such an ILP system would be able to harmoniously combine relations with "numerical methods" in the same *model.* A proper approach to deal with numerical domains would therefore extend the applicability of ILP systems.

Current ILP approaches [4] to numerical domains usually carry out a search through the model (hypothesis) space looking for a minimal value of a cost function like the Root Mean Square Error (RMSE). Systems like TILDE [5] are of that kind. One problem with the minimisation of RMSE in noisy domains is that the models tend to be brittle. The error is small when covering a small number of examples. The end result is a large set of clauses to cover the complete set of examples. This is a drawback on the intelligibility of ILP induced models. This aspect is also an obstacle to the induction of a *numerical theory,* since we end up with small locally fitted *sub-models,* that may not correspond to the overall structure of the underlying process that generated data.

In this paper we introduce improvements on the numerical reasoning capabilities of ILP systems by adopting statistical-based noise handling techniques such as: (i) model validation and; (ii) model selection. We also propose a new stopping criterium based on the PAC [6] method to evaluate learning performance. The proposals lead to considerable improvements in the ILP system used on the empirical evaluation. The experimental results show that an ILP system extended with such procedures compares very well with statistical methods.

The rest of the paper is organised as follows. Section 2 identifies the steps of a basic ILP algorithm that are subject to improvements proposed in this paper. The proposals for Model Validation are discussed in Section 3. In Section 4 we propose the the stopping criterium. The proposals for Model Selection are discussed in Section 5. Section 6 presents the experimental findings. The related work is discussed in Section 7. Finally, in Section 8 we draw the conclusions.

2 Search Improvements

In ILP, the search procedure is usually an iterative greedy set-covering algorithm that finds the best clause on each iteration and removes the covered examples. Each hypothesis generated during the search is evaluated to determine their quality. A widely used approach in classification tasks is to score a hypothesis by measuring its coverage. That is, the number of examples it explains. In numerical domains it is common to use the RMSE or Mean Absolute Error (MAE) as a score measure. Algorithm 1 presents an overview of the procedure and identifies the steps modified by our proposals.

Algorithm 1 Basic cycle of a greedy set-covering ILP algorithm

```
1: repeat
2:     repeat
3:         synthesise a hypothesis
4:         accept a hypothesis (Model Validation)
5:         Update best hypothesis (Model Selection)
6:     until Stopping Criterion satisfied (PAC-Based Stopping Criterium)
7:     Remove explained examples
8: until "All" examples explained
```

We propose an improvement to step 4 where a hypothesis is checked if it is a satisfactory approximation of the underlying process that generated data. We propose the use of statistical tests in that model validation[1] step. Step 5 is improved to mitigate the over-fitting problem, which the fragmented structure

[1] Model Validation term is used in the statistics sense.

of the induced theories suggests, using a model selection criterium. Our proposal for step 6 is inspired on the PAG [6] method to evaluate learning performance.

We use the terms hypothesis, model and theory with the following meaning in this paper. An hypothesis is a conjecture after a specific observation and before any empirical evaluation has been performed. A model is a hypothesis that has at least limited validity for predicting new observations. A model is a hypothesis that has passed the model validation tests. A Theory is a set of hypotheses whose prediction capabilities have been confirmed through empirical evaluation.

3 Model Validation

In most applications, the true nature of the model is unknown, therefore, it is of fundamental importance to assess the goodness-of-fit of each conjectured hypothesis. This is performed in a step of the induction process called *Model Validation.* Model Validation allows the system to check if the hypothesis is indeed a satisfactory model of the data. This step is common both in Machine Learning and Statistical Inference.

There are various ways of checking if a model is satisfactory. The most common approach is to examine the residuals. The residuals are the random process formed from the differences between the observed and predicted values of a variable. The behaviour of the residuals may be used to check the adequacy of the fitted model as a consequence of the Wold's theorem, defined as follows.

Theorem 1 (Wold's Theorem). *Any real-valued stationary process may be decomposed into two different parts. The first is totally deterministic. The second totally stochastic. The stochastic part of the process may be written as a sequence of serially uncorrelated random variables z with zero mean and variance σ^2. The stationarity condition imply $\sigma < \infty$, thus z is a White Noise (WN) process:*

$$z \sim WN(0, \sigma) \tag{1}$$

According to condition (1) of the Wold's theorem, if the fitted model belongs to the set of "correct" functional classes, the residuals should behave like a white noise process with zero mean and constant variance. Hypotheses whose residuals do not comply with condition (1) may be rejected using specific statistical tests that check randomness. The Ljung-Box test [7], is one of such tests. The null hypothesis of the Ljung-Box test is a strict white noise process. Thus, residuals are independent and identically distributed (i.i.d.). According to the definition of statistical independence residuals are incompressible. Muggleton and Srinivasan [8], have also proposed to check noise incompressibility for evaluating hypothesis significance but in the context of classification problems.

Other statistical tests may be incorporated to check our assumptions regarding error structure, like tests for normality. The use of residuals for model assessment is a very general method which apply to many situations.

4 Stopping Criterium

The stopping criterium proposed is based on the PAC [6] method to evaluate learning performance: $P(|z| > \epsilon) < \delta$. The stopping criterium stops the search whenever the probability that the error is greater than the accuracy (ϵ) is less than the confidence interval (δ). Different degrees of "goodness" will correspond to different values of ϵ and δ.

In this section we propose to calculate the bound, δ, for any unknown distribution. Theorem 2, proves the existence of that bound for a single clause (hypothesis) and provides a procedure to calculate the error probability for a given accuracy level. Corollary 1, generalises the bound on the error probability to multi-clausal theories.

Theorem 2 (Bounding Error Probability of an Hypothesis). *Let z be the residuals from the hypothesis h_i. Assume z is independent and identically distributed (i.i.d.) with distribution variance σ^2. Then the probability of the error being greater then ϵ is bounded by:*

$$P(|z| > \epsilon \mid h_i) < \delta, \quad \delta = \frac{\sigma^2}{\epsilon^2} \tag{2}$$

Proof: Let the residuals $z_1, z_2, \ldots, z_n$ be a sequence of i.i.d. random variables each with finite mean μ and σ. if $\bar{z} = (z_1 + \ldots + z_n)/n$ is the average of $z_1, z_2, \ldots, z_n$, then, it follows from the week law of large numbers [7] that:

$$\bar{z} \xrightarrow{P} \mu \tag{3}$$

Let the sample variance be $S_n = \frac{1}{n}\sum_{j=1}^{n}(z_j - \bar{z})^2 = \frac{1}{n}\sum_{j=1}^{n} z_j^2 - \bar{z}^2$, where $\bar{z}$ is the sample average. It follows from the Slutski's lemma [7] that:

$$S_n \xrightarrow{P} \sigma \tag{4}$$

Assuming the residuals z of the hypothesis h_i pass the null hypothesis of the Ljung-Box test, then they will comply with a strict white noise process with zero mean and finite variance, yielding thereby:

$$\mu = 0, \qquad \sigma < \infty \tag{5}$$

Following Conditions (3) and (4), each observation may be considered drawn from the same ensemble distribution. Thus, the sample mean and variance of the joint distribution converge to the ensemble mean and variance. Moreover, condition (5) states that both values are finite and, therefore, for all $\epsilon > 0$, the Chebishev's inequality bounds the probability of the residuals value, z, being greater then ϵ to:

$$P(|z| > \epsilon \mid h_i) < \frac{\sigma^2}{\epsilon^2} \tag{6}$$

■

Corollary 1 (Bounding Error Probability of a Theory). *Let H be a set of hypothesis (clauses) that describes a given theory T. Assume:*

$$P(|z| > \epsilon \mid h_i) < \delta, \quad \forall_{h_i \in H} \tag{7}$$

then, for theory T, the probability of the error, z, being greater than ϵ, is also bounded by $P(|z| > \epsilon) < \delta$.

We recall that just one clause is activated at each time thus all clauses of a non-recursive theory are mutually exclusive regarding example coverage, i.e.

$$h_i \cap h_j = \emptyset \quad \forall_{i \neq j} \tag{8}$$

We also recall that the prior probability of h_i, $P(h_i)$ may be estimated calculating the frequency of h_i on the training set and dividing it by the coverage of the theory. Because the sum of the frequencies of all hypotheses is equal to the theory coverage, then

$$\sum_{\forall_{h_i \in H}} P(h_i) = 1 \tag{9}$$

Proof: Let conditions (8) and (9) hold, then it follows from the *total probability theorem* that:

$$P(|z| > \epsilon) = \sum_{\forall_{h_i} \in H} P(|z| > \epsilon \mid h_i) P(h_i). \tag{10}$$

Let condition (7) hold, then we may substitute $P(|z| > \epsilon \mid h_i)$ by δ in equation (10), yielding thereby: $P(|z| > \epsilon) < \delta \sum_{\forall_{h_i \in H}} P(h_i)$. Since $\sum_{\forall_{h_i \in H}} P(h_i) = 1$ and $P(|z| > \epsilon \mid h_i) < \delta, \quad \forall_{h_i \in H}$, then:

$$P(|z| > \epsilon) < \delta \tag{11}$$

■

5 Model Selection

The evaluation of conjectured hypotheses is central to the search process in ILP. Given a set of hypotheses of the underlying process that generated data, we wish to select the one that best approximates the "true" process. The process of evaluating candidate hypothesis is termed *model selection* in statistical inference.

A simple approach to model selection is to choose the hypothesis that gives the most accurate description of the data. For example, select the hypothesis that minimises RMSE. However, model selection is disturbed by the presence of noise in data, leading to the problem of over fitting. Thus, a hypothesis with larger number of adjustable parameters has more flexibility to capture complex structures in data but also to fit noise. Hence, any criterium for model selection should establish a trade-off between descriptive accuracy and hypothesis complexity.

5.1 Hypothesis Complexity

Defining a theoretically well-justified measure of model complexity is a central issue in model selection that is yet to be fully understood. In Machine Learning, some authors have advanced their own definition of complexity. Muggleton [9] proposes a complexity measure based on the number of bits necessary to encode a hypothesis. Dzerovski [10], proposes a complexity measure based on the length of a grammar sentence in the Lagramge system. Both of the referred complexity measures are sensitive to the hypothesis functional form. This is clear since both penalise each literal added. We argue that the functional form is not a good approximation to measure the complexity of a real-valued hypothesis, since any real-valued function can be accurately approximated using a single functional class. This follows directly from Approximation Theory as the Kolmogorov's superposition theorem illustrates.

Theorem 3 (Kolmogorov Superposition Theorem). *Any continuous multidimensional function $f(x_1, \ldots, x_m)$, can be represented as the sum of $m+1$ functions. These functions are called universal functions because depend only on the dimensionality m and not in the functional form of f.*

Following theorem 3, the sum of universal functions is proportional to the dimensionality m. This highlights the role of dimensionality on a definition of hypothesis complexity. A few arguments on computational complexity and estimation theory also support this claim. Since the machine learning algorithm is given a finite dataset, models with fewer adjusted parameters will be easier to optimise since they will generically have fewer misleading local minima in the error surfaces associated with the estimation. They will be also less prone to the curse of dimensionality. They will require less computational time to manipulate. A model with fewer degrees of freedom generically will be less able to fit statistical artifacts in small data sets and will therefore be less prone to the so-called "generalisation error". Finally, several authors (Akaike [11]; Efron [12]; Ye [13]) proposed measures of model complexity which in general depend on the number of adjusted parameters. Consequentially, the adopted measure of complexity in this work is the number of adjusted parameters to data.

5.2 Model Selection Criteria

There are several model selection criteria considering the adopted measure of model complexity. Among these, we may find: (i) Akaike Information Criterium (AIC) [11], defined as $AIC = -2\ln(L) + 2k$; (ii) Akaike Information Criterium Corrected for small sample bias(AICC) [11], defined as $AICC = -2\ln(L) + 2k\frac{n}{n-k+1}$; (iii) Bayesian Information Criterium (BIC) [14], defined as $BIC = -\ln(L) + \ln(n)k$ and; (iv) the Minimum Description Length (MDL) [11], defined as $MDL = -\ln(L) + \frac{k}{2}\ln(n) + (\frac{k}{2}+1)\ln(k+1)$.

The estimation of an hypothesis likelihood function, L, with k adjusted parameters, requires a considerable computational effort and the assumption of prior distributions. In this context, the Gaussian distribution plays an important role in the characterisation of the noise, fundamentally due the central limit theorem. Assuming error is i.i.d. drawn from a Gaussian distribution then, the likelihood of a hypothesis given the data [11] is: $\ln(L) = -\frac{n}{2}(1+\ln(2\pi)+\ln(\hat{\sigma_r}^2))$, where $\hat{\sigma}_r^2 = \frac{1}{n}\sum_{i=1}^{n} z_i^2$, and z are the residuals of the induced hypothesis. Since the compared hypotheses may have different coverages, the criteria were normalised by the sample size, as suggested by Box and Jenkins [15](pg. 201). This approach has the advantage of indirectly biasing the search to favour hypotheses with higher coverage, and consequentially, theories with less clauses.

Analytical model selection criteria like AIC and BIC are asymptotically equivalent to leave-one-out and leave-v-out cross-validation [16]. However, they have the advantage of being incorporated in the cost function, which reduces relatively the computational effort.

Other authors presented similar work in this area. Zelezni [17] derives a model selection criterium under similar assumptions that uses the Muggleton's complexity measure, which according to the adopted definition of complexity, is unsuitable for our purposes. It also requires the calculation of the "generality" function for each induced hypothesis. His formulation does not estimate the modal value of the likelihood, so the final equation includes the usually unknown nuisance parameter of the hypothesis, which somehow limits its practical use.

5.3 Choosing a Model Selection Criterium

Model selection criteria have different characteristics, thus, it is essential to clarify its application conditions to numerical problems in ILP.

The use of AIC is recommendable if the data generating function is not in any of the candidate hypotheses and if the number of models of the same dimension does not grow very fast in dimension, then the average squared error of the selected model by AIC is asymptotically equivalent to the smallest possible one offered by the candidate models [16]. Otherwise, AIC cannot be asymptotically optimal, increasing model complexity as more data is supplied [14].

The use of BIC and other dimension consistent criteria like MDL is advisable if the correct models are among the candidate hypothesis, then the probability of selecting the true model by BIC approaches 1 as $n \to \infty$. Otherwise, BIC has a bias for choosing oversimplified models [16].

Since in machine learning except rare instances the true model is never known or when it is known is usually intractable, AIC-like criteria are preferable.

6 Experimental Evaluation

This section presents empirical evidence for the proposals made in this paper. We propose to use an ILP system to learn a new time series model for prediction. In this sense, the experiment has been inspired in the Colton and Muggleton [18] application of an ILP system to scientific discovery tasks. It is also related with Zelezni's [17] work since it learns a numeric function. The datasets used were collected from statistics and time series literature. For each dataset we compared our results with the published ones.

6.1 Datasets

Canada's Industrial Production Index [19]; USA Unemployment rate [20]; ECG of a patient with sleep apnea [21] and; VBR Traffic of an MPEG video [22]. These datasets consist of facts that relate the time with an output variable. The time is expressed in discrete intervals and the output is a real-valued variable. The mode declaration for the head literal is of the form: timeseries(+Time, –Output).

6.2 Benchmark Models

In this experiment we compared theories induced by the raw IndLog system with the following models: IndLog with Model Validation and Model Selection activated (IndLog$^{\text{MVS}}$); Auto-Regressive Integrated Moving Average (ARIMA); Threshold Auto-Regressive (TAR); Markov Switching Autoregressive (MSA); Autoregressive model with multiple structural Changes (MSC); Self-Excited Threshold Auto-Regressive (SETAR); Markov Switching regime dependent Intercepts Autoregressive parameters and (H)variances(MSIAH); Markov Switching regime dependent Means and (H)variances (MSMH); Bivariate Auto-Regressive models(Bivariate AR) and; Radial Basis Functions Networks (RBFN).

6.3 Learning Task Description

The experiment consists of learning a modified class of the TAR [23] model using an ILP system. The model extends the TAR model , by adding extra degrees of freedom that will be estimated in run-time. The induced clauses are of the kind: $\text{timeseries}(T, X) \leftarrow \text{inInterval}(R_m, T, D, R_{m+1}), \text{armodel}(T,P,X)$. Where the variable X is the time-series observed, m is the index denoting the sub-region, R_m denotes the threshold amplitude of region m and *armodel* is the background knowledge literal for the AR Model [23]. The main difference from the original TAR structure is that instead of a single D value, we have one D for each sub-region. The learning task consists of estimating: (i) the number of parameters p of each AR sub-model; (ii) The time delay D of each sub-model and; (iii) The thresholds that bound each subregion R_m and R_{m+1}.

6.4 Results Summary and Discussion

This section presents the results obtained for each dataset of Section 6.1. Those datasets were studied in several papers, using different classes of models. Thus, all time series datasets in table 1 have an AR model that may be used as a reference across datasets. The recall number for the Unemployment, Production, VBR Traffic, and ECG datasets are respectively: 100%, 96%, 94%, and 78%.

The ILP system consistently induced models with best forecasting performance on all datasets studied. This allow us to conclude that the proposed modifications to the basic ILP search process, makes an ILP system suited for time series forecasting and discovering new time series models.

Table 1. Summary of results of the Relative RMSE of the ILP algorithm and other benchmark models for the selected datasets

Model	Unemployment	Production	VBR Traffic	ECG
IndLog^{MVS}	0.91	0.85	0.93	0.82
IndLog	1.11	1.04	0.96	0.93
AR	1.04	0.98	0.94	0.97
SARIMA	1.00	1.00	-	-
MSA	1.19	-	-	-
MSC	-	1.00	-	-
MSMH	-	0.98	-	-
MSIAH	-	1.20	-	-
SETAR	-	1.19	-	-
TAR	1.00	-	1.00	-
RBFN	-	-	-	1.00
Bivariate AR	1.20	-	-	-
Benchmark RMSE	1.59E-1	4.44E-3	12.93E3	4.53

7 Related Work

Other approaches to the task of learning numerical relationships may be found in the ILP literature. FORS [24] integrates feature construction into linear regression modelling. The ILP system IndLog [25] presented mechanisms for coping with large number of examples that include noisy numerical data without negative examples and the capability to adjust model parameters at induction time. Equation discovery systems like LAGRAMGE [10] allow the user to specify the space of possible equations using a context-free grammar. TILDE [5] has the capability of performing regression-like tasks.

8 Conclusions

In this paper we have proposed improvements in the numerical reasoning capabilities of ILP systems. The improvements proposed are: model validation; model selection criteria and; a stopping criterium.

Our proposals were incorporated in the IndLog [25] ILP system and evaluated on time series modelling problems. The ILP results were better than other statistics-based time series prediction methods. The ILP system discovered a new switching model based on the possibility of varying the delay on the activation rule of each sub-model of a TAR model.

The proposals made for model validation, model selection and for measuring the learning performance can be generalised to other machine learning techniques dealing with numerical reasoning.

References

1. S. Muggleton. Inductive logic programming. *New Generation Computing,* 8(4):295–317, 1991.
2. R.D. King, S. Muggleton, A. Srinivasan, and M. Sternberg. Structure-activity relationships derived by machine learning. *Proceedings of the National Academy of Sciences,* 93:438–442, 1996.
3. J. Cussens. Part-of-speech tagging using progol. In *In* Proc. of the 7th Int. Workshop on Inductive Logic Programming, pages 93–108. Springer, 1997.
4. A. Srinivasan and R. Camacho. Numerical reasoning with an ILP system capable of lazy evaluation and customised search. *J. Logic Prog.,* 40(2-3):185–213, 1999.
5. Hendrik Blockeel, Luc De Raedt, and Jan Ramon. Top-down induction of clustering trees. In J. Shavlik, editor, *15th ICML,* pages 55–63. Morgan Kaufmann, 1998.
6. L. Valiant. A theory of the learnable. *Commun. ACM,* 27(11):1134–1142, 1984.
7. P. Brockwell and R. Davis. *Time series: theory and methods.* Springer, N.Y., 1991.
8. S. Muggleton, A. Srinivasan, and M. Bain. Compression, significance and accuracy. In D. et al. Sleeman, editor, *ML92,* pages 338–347. Morgan Kauffman, 1992.
9. S. Muggleton. Learning from positive data. In S. Muggleton, editor, *ILP96,* volume 1314 of *LNAI,* pages 358–376. Springer, 1996.
10. S. Dzeroski and L. Todorovski. Declarative bias in equation discovery. In *14th ICML,* pages 376–384, USA, 1997.
11. K. Burnham and D. Anderson. *Model Selection and Multimodel Inference.* Springer, New York, 2002.
12. B. Efron. How biased is the apparent error rate of a prediction rule? *JASA,* 81:461–470, 1986.
13. J. Ye. On measuring and correcting the effects of data mining and model selection. *JASA,* pages 120–131, 1998.
14. Schwarz. Estimating the dimension of a model. *Annals of Statistics,* 6:461–464, 1978.
15. Jenkins Box and Reinsel. *Time Series Analysis, Forecasting and Control.* Prentice Hall, Englewood Cliffs, N.J., USA, 3rd edition edition, 1994.
16. J Shao. An asymptotic theory for linear model selection. *Statistica Sinica,* 7:221–264, 1997.
17. Filip Zelezny. Learning functions from imperfect positive data. In *International Workshop on Inductive Logic Programming,* pages 248–260, 2001.
18. S. Colton and S. Muggleton. ILP for mathematical discovery. In T. Horváth and A. Yamamoto, editors, *ILP03,* volume 2835 of *LNAI,* pages 93–111. Springer, 2003.
19. B. Silvertovs and D. Dijk. Forecasting industrial production with linear, nonlinear and structural change models. Technical report, Erasmus University, 2003.
20. R. Tsay l. Montgomery, V. Zarnowitz and G. Tiao. Forecasting the u.s. unemployment rate. *JASA,* 93:478–493, 1998.
21. M. et al Kreutz. Structure optimization of density estimation models applied to regression problems. In *Proc. of the 7th Int. Workshop on AI and Statistics,* 1999.
22. B. Jang and C. Thomson. Threshold autoregressive models for vbr mpeg video traces. In *IEEE INFOCOM,* pages 209–216, USA, 1998.
23. H. Tong. *Nonlinear time series, a dynamical system approach.* Claredon press, Oxford, UK, 1st edition edition, 1990.
24. A. Karalic and I. Bratko. First order regression. *Machine Learning,* 26:147–176, 1997.
25. R. Camacho. *Inducing Models of Human Control Skills using Machine Learning Algorithms.* PhD thesis, Universidade do Porto, July 2000.

Enhanced ICA Mixture Model for Unsupervised Classification

Patrícia R. Oliveira and Roseli A. F. Romero

Department of Computer Science and Statistics,
Institute of Mathematics and Computer Science – University of São Paulo,
Av. Trabalhador Sancarlense,
400 Caixa Postal 668, 13560-970, São Carlos, SP, Brazil
{rufino, rafrance}@icmc.usp.br

Abstract. The ICA mixture model was originally proposed to perform unsupervised classification of data modelled as a mixture of classes described by linear combinations of independent, non-Gaussian densities. Since the original learning algorithm is based on a gradient optimization technique, it was noted that its performance is affected by some known limitations associated with this kind of approach. In this paper, improvements based on implementation and modelling aspects are incorporated to ICA mixture model aiming to achieve better classification results. Comparative experimental results obtained by the enhanced method and the original one are presented to show that the proposed modifications can significantly improve the classification performance considering random generated data and the well-known iris flower data set.

1 Introduction

Classification and grouping of patterns are problems commonly found in many branches of science such as biology, medicine, computer vision and artificial intelligence [7]. In a supervised classification, each pattern in a data set is identified as a member of a predefined class. On the other hand, an unsupervised classification algorithm assigns each pattern to one class based on statistics only, without any knowledge about training classes.

An approach for unsupervised classification is based on mixture models (see, for example, [5]) where the data distribution is modelled as a weighted sum of class-conditional densities. For example, in the case of a Gaussian mixture model, the data in each class is assumed to have a multivariate Gaussian distribution. However, this assumption implies that the Gaussian mixture model exploits only second order statistics (mean and covariance) of the observed data to estimate the posterior densities.

In recent years, Independent Component Analysis (ICA) has been widely applied in many research fields due the fact that it exploits higher order statistics in data [6] [4] [2]. In fact, ICA is a generalization of Principal Component Analysis (PCA), in the sense that ICA transforms the original variables into independent components instead of just uncorrelated ones, as in the case of PCA.

The ICA Mixture Model (ICAMM) was proposed by Lee and coworkers in [9], aiming to overcome one limitation of ICA, that is the assumption that the sources are independent. In such approach, this assumption was relaxed by using the concept of mixture models.

Each class in ICAMM is described by a linear combination of independent sources with non-Gaussian densities. The algorithm finds independent components and the mixing matrix for each class and also computes the class membership probability for each pattern in the data set. The learning rules for ICAMM were derived using gradient ascent to maximize the log-likelihood data function.

Although some promising features of ICAMM have been reported in [9], we observed, in experiments with random generated data and Iris data set, a very slow convergence and poor classification results. In two papers found in literature [11], [12], ICAMM was also applied without great success. In [11], ICAMM was shown to work well on 2D artificial data, but the advantage of using such model in image segmentation was not proven conclusively. In [12] some feature extraction techniques were considered as preprocessing steps for reducing the data dimensionality and increasing the efficiency of ICAMM. Although the mean overall classification accuracies obtained by their approach are higher than those obtained by the k-means method, they pointed out several limitations and assumptions that compromise the use of ICAMM in remote sensing classification.

In attempting to improve the ICAMM performance, this work introduces the Enhanced ICA Mixture Model (EICAMM), which implements some modifications on the original ICAMM algorithm learning rules. As in the original ICAMM, some modifications are also based on an information-maximization approach proposed by Bell and Sejnowski in [2]. In such paper, they introduce a new self-organizing algorithm which maximizes the information transferred in a network of non-linear units.

Another problem associated to the ICAMM is related to the fact that its learning algorithm is based on a gradient optimization technique. Therefore, it was noted that its performance is affected by some known limitations associated with this kind of approach. The gradient ascent (descent) algorithm has became famous in literature as a standard method of training neural networks [10]. The widespread use of such technique is mainly related to its most powerful property: it can be mathematically proven that this algorithm will always converge to a local minimum in the objective function, although an immense number of iterations are often necessary. Beside this point, another important problem to be solved is that there is no guarantee that the method will not be stuck in a local minimum.

Aiming to attenuate these problems, additional improvements on the original model were made by incorporating some features of methods of nonlinear optimization based on second derivatives of objective functions. In the sense, the Levenberg-Marquardt method (see for example [10]), was incorporated to the learning algorithm to guarantee and improve the convergence of the model.

In order to evaluate the efficiency of the proposed modifications, some results obtained by the Enhanced ICA Mixture Model (EICAMM) and those obtained by the original one are presented and discussed in a comparative study. From this discussion, one can see that the proposed modifications can significantly improve the classification performance, considering random generated data and the well-known Iris flower data set.

This paper is organized as follows. In section 2, basis concepts on ICA are presented. The original ICA mixture model is presented in section 3. In section 4, the Enhanced ICA Mixture Model is introduced. Experimental results are presented and discussed in Section 5. Finally, some conclusions and future work are presented in Section 6.

2 Independent Component Analysis (ICA)

The noise-free ICA model can be defined as the following linear model:

$$\mathbf{x} = \mathbf{A}\mathbf{s} \tag{1}$$

where the n observed random variables $x_1, x_2, ..., x_n$ are modelled as linear combinations of n random variables $s_1, s_2, ..., s_n$, which are assumed to be statistically mutually independent.

In such model, the independent components s_j cannot be directly observed and the mixing coefficients a_{ij} are also assumed to be unknown. Only the random variables x_i are observed and both the components s_j and the coefficients a_{ij} must be estimated using $\mathbf{x}$.

The independent components s_j must have non-Gaussian distributions. Kurtosis is commonly used as measure of non-Gaussianity in ICA estimation of original sources. Such measure for a random variable y is given by:

$$\text{kurt}(y) = E\{y^4\} - 3 \tag{2}$$

A Gaussian variable has a zero kurtosis value. Random variables with positive kurtosis have a super-Gaussian distribution and those with negative kurtosis a have sub-Gaussian distribution, as illustrated in Figure 1.

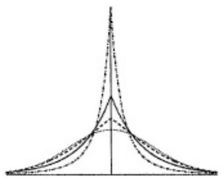


Fig. 1. Examples of Gaussian, super-Gaussian and sub-Gaussian probability density functions. Solid line: Laplacian density. Dashed line: a typical moderately super-Gaussian density. Dash-dotted line: a typical strongly super-Gaussian density. Dotted line: Gaussian density [6]

It is also assumed that the unknown mixture matrix $\mathbf{A}$ is square. It simplifies the estimation process, since after estimating matrix $\mathbf{A}$, it is possible to compute its inverse $\mathbf{W}$, and obtain the independent components simply by:

$$\mathbf{s} = \mathbf{W}\mathbf{x} \tag{3}$$

Therefore, the goal of ICA is to find a linear transformation **W** of sensor signals **x** that makes the outputs **u** as independent as possible

$$\mathbf{u}_t = \mathbf{W}\mathbf{x}_t = \mathbf{W}\mathbf{A}\mathbf{s}_t \tag{4}$$

so that **u** in an estimative vector of the sources.

There are several methods for adapting the mixing mixtures in the ICA model [4], [3], [6] [8]. In [8], they use the extended infomax ICA learning rule to blindly separate unknown sources with sub-Gaussian and super-Gaussian distributions. In this approach, maximizing the log-likelihood of the data **X** gives the following learning rule for **W**:

$$\Delta\mathbf{W} = [\mathbf{I} - \varphi(\mathbf{u})\mathbf{u}^T]\mathbf{W}, \tag{5}$$

where **I** is the identity matrix and

$$\varphi(\mathbf{u}) = -\frac{\frac{\partial p(\mathbf{u})}{\partial u}}{p(\mathbf{u})} = \left[-\frac{\frac{\partial p(u_1)}{\partial u_1}}{p(u_1)}, \ldots, -\frac{\frac{\partial p(u_N)}{\partial u_N}}{p(u_N)}\right]. \tag{6}$$

A strictly symmetric sub-Gaussian bimodal distribution can be modelled by a weighted sum of two Gaussian distributions with mean μ and variance σ^2, given as:

$$p(\mathbf{u}) = \frac{1}{2}(N(\mu, \sigma^2) + N(-\mu, \sigma^2)), \tag{7}$$

For $\mu = 1$ and $\sigma^2 = 1$, Equation(6) reduces to:

$$\varphi(\mathbf{u}) = \mathbf{u} - \tanh(\mathbf{u}), \tag{8}$$

which leads to the learning rule for strictly sub-Gaussian sources:

$$\Delta\mathbf{W} \propto [\mathbf{I} + \tanh(\mathbf{u})\mathbf{u}^T - \mathbf{u}\mathbf{u}^T]\mathbf{W}. \tag{9}$$

For unimodal super-Gaussian sources, the following density model is adopted:

$$p(\mathbf{u}) = N(0, 1)\mathrm{sech}^2\mathbf{u}, \tag{10}$$

In this case, Equation(6) reduces to:

$$\varphi(\mathbf{u}) = \mathbf{u} + \tanh(\mathbf{u}), \tag{11}$$

which leads to the following learning rule for strictly super-Gaussian sources:

$$\Delta\mathbf{W} \propto [\mathbf{I} - \tanh(\mathbf{u})\mathbf{u}^T - \mathbf{u}\mathbf{u}^T]\mathbf{W}. \tag{12}$$

Therefore, using the Equations (9) and (12), one can obtain a generalized learning rule, using the switching criterion for distinguishing between sub-Gaussian sources and super-Gaussian sources by the sign before the hyperbolic tangent function as:

$$\Delta\mathbf{W} \propto [\mathbf{I} - \mathbf{K}\tanh(\mathbf{u})\mathbf{u}^T - \mathbf{u}\mathbf{u}^T]\mathbf{W}. \tag{13}$$

where $\mathbf{K}$ is an N-dimensional diagonal matrix composed of k's calculated as:

$$k_i = \text{sign}(\text{kurt}(u_i)). \tag{14}$$

3 The ICA Mixture Model (ICAMM)

In ICAMM, it is assumed that the data $\mathbf{X} = \{\mathbf{x}_1, \dots \mathbf{x}_T\}$ are generated by a mixture density model. The likelihood of data is given by the joint density

$$p(\mathbf{X} \mid \theta) = \prod_{t=1}^{T} p(\mathbf{x}_t \mid \theta). \tag{15}$$

The mixture density is

$$p(\mathbf{x}_t \mid \theta) = \sum_{k=1}^{K} p(\mathbf{x}_t \mid C_k, \theta_k) p(C_k), \tag{16}$$

where $\theta = (\theta_1, \dots, \theta_K)$ are the unknown parameters for each $p(\mathbf{x} \mid C_k, \theta_k)$. In this case, C_k denotes the class k and it is assumed that the number of classes k is known in advance. The data in each class are described by:

$$\mathbf{x}_t = \mathbf{A}_k \mathbf{s}_k + \mathbf{b}_k \tag{17}$$

where $\mathbf{A}_k$ is a $N \times N$ scalar matrix and $\mathbf{b}_k$ is the bias vector for class k. The vector $\mathbf{s}_k$ is called the source vector.

The task is to classify the unlabelled data points and to determine the parameters for each class $(\mathbf{A}_k, \mathbf{b}_k)$ and the probability for each data point.

The iterative learning algorithm which performs gradient ascent on the total likelihood of data has the following steps:

1. Compute the log-likelihood of data for each class:

$$\log p(\mathbf{x}_t \mid C_k, \theta_k) = \log p(\mathbf{s}_k) - \log(\mid \det(\mathbf{A}_k) \mid) \tag{18}$$

   In ICAMM, $\mathbf{s}_k = \mathbf{A}_k^{-1}(\mathbf{x}_t - \mathbf{b}_k)$.
2. Compute the probability for each class given the data vector $\mathbf{x}_t$:

$$p(C_k \mid \mathbf{x}_t, \theta) = \frac{p(\mathbf{x}_t \mid C_k, \theta_k) p(C_k)}{\sum_k p(\mathbf{x}_t \mid C_k, \theta_k) p(C_k)} \tag{19}$$

3. Adapt the matrices $\mathbf{A}_k$ and the bias terms $\mathbf{b}_k$ using the updating rules:

$$\Delta \mathbf{A}_k \propto -p(C_k \mid \mathbf{x}_t, \theta) \mathbf{A}_k [\mathbf{I} - \mathbf{K} \tanh(\mathbf{s}_k) \mathbf{s}_k^T - \mathbf{s}_k \mathbf{s}_k^T] \tag{20}$$

$$\mathbf{b}_k = \frac{\sum_t \mathbf{x}_t p(C_k \mid \mathbf{x}_t, \theta)}{\sum_t p(C_k \mid \mathbf{x}_t, \theta)} \tag{21}$$

For log-likelihood estimation in Equation 18, $\log p(\mathbf{s}_k)$ is modelled by:

$$\log p(\mathbf{s}_k) \propto -\sum_{i=1}^{N} \left(k_i \log(\cosh s_{k,i}) - \frac{s_{k,i}^2}{2} \right) \tag{22}$$

The learning algorithm converges when the change in log-likelihood function between two successive iterations is below than a predetermined small constant. After the convergence, the classification process is carried by processing each instance with the learned parameters $\mathbf{A}_k$ and $\mathbf{b}_k$. The class-conditional probabilities $p(C_k \mid \mathbf{x}_t, \theta)$ are computed for all classes the corresponding instance label is assigned to the class with highest conditional probability.

4 Enhanced ICA Mixture Model (EICAMM)

The Enhanced ICA Mixture Model (EICAMM) proposed in this paper was derived from some important modifications on ICAMM, considering both modelling and implementation aspects, which are discussed in this section.

4.1 Reformulating the Class Model

Instead of considering the bias term as added to the data after they are generated by an ICA model (Equation 17), in EICAMM, the bias vectors are considered to be mixed to the signal sources. This modification originates another equation to describe the data in each class, as given by:

$$\mathbf{x}_t = \mathbf{A}_k(\mathbf{s}_k + \mathbf{b}_k) \tag{23}$$

4.2 Reformulating the Bias Learning Rule

For a more informative bias learning rule, in EICAMM, the bias updating rule is derived as in [2], where it is formulated by using an approach to maximize the mutual information that the output Y of a neural network processor contains about its input X. According to the ideas in [2], when a single input x passes through a nonlinear transformation function $g(x)$, yields an output variable y, in a way that the mutual information between these two variables is maximized, aligning high density parts of the probability density function of x with highly sloping parts of the function $g(x)$.

In this sense, the learning rule for weights and biases in such neural network are formulated using some nonlinear transfer function. In the formulation of EICAMM, the learning rule for the bias term is formulated as in [2], considering the hyperbolic tangent transfer function, as given by:

$$\Delta \mathbf{b}_k \propto -2\tanh(\mathbf{s}_k). \tag{24}$$

4.3 An Orthogonalization Problem

Since from the derived formulation of ICAMM [9], $\mathbf{W}_k = \mathbf{A}_k^{-1}$, the Equation (13) can be written as:

$$\Delta\mathbf{W} \propto [\mathbf{I} - \mathbf{K}\tanh(\mathbf{u})\mathbf{u}^T - \mathbf{u}\mathbf{u}^T]\mathbf{A}^{-1}. \tag{25}$$

Supposing that $\mathbf{A}$ is an orthogonal matrix then, by a propriety of orthogonality, one have that $\mathbf{A}^{-1} = \mathbf{A}^T$. Therefore, as one more modification that was incorporated to EICAMM, the updating rule of $\mathbf{A}_k$ is now given by:

$$\Delta\mathbf{A}_k \propto p(C_k \mid \mathbf{x}_t, \theta)\mathbf{A}_k(\mathbf{I} - \mathbf{K}\tanh(\mathbf{s}_k)\mathbf{s}_k^T - \mathbf{s}_k\mathbf{s}_k^T)^T, \tag{26}$$

where $\mathbf{s}_k = \mathbf{A}_k^T\mathbf{x}_t - \mathbf{b}_k$, considering the model for the data in each class used in EICAMM and formulated in Equation (23). Note that that the transpose operator is used instead of the inverse matrix $\mathbf{A}_k^{-1}$, which implies a computational advantage of EICAMM in comparison to ICAMM.

Consequently, in EICAMM, the mixing matrices $\mathbf{A}_k$ should be orthogonalized in each iteration using the following equation:

$$\mathbf{A}_k = \mathbf{A}_k(\mathbf{A}_k^T\mathbf{A}_k)^{1/2}. \tag{27}$$

4.4 Incorporating Second Derivative Information

When the second derivative of the objective function is relatively easy to compute, it is considered an excellent idea to incorporate this information for speeding up the convergence and bounding the minimum of the function. Following this motivation, a modification in the updating rule for $\mathbf{A}_k$ has been proposed here which incorporates the second derivative of the log-likelihood function. In this section, it is presented how the Newton's method and the Levenberg-Marquardt method can be formalized to be used in EICAMM.

• **Newton's Method:** For modelling the Newton's method for EICAMM, the second derivatives of the log-likelihood, which is given by the Hessian matrix **H**, should be incorporated to the learning rule in Equation (20). Consequently, the new updating rule is given by:

$$\Delta\mathbf{A}_k \propto p(C_k \mid \mathbf{x}_t, \theta)\mathbf{H}^{-1}\mathbf{A}_k(\mathbf{I} - \mathbf{K}\tanh(\mathbf{s}_k)\mathbf{s}_k^T - \mathbf{s}_k\mathbf{s}_k^T)^T. \tag{28}$$

The derivation of matrix **H** for EICAMM is presented in Appendix A.

• **Levenberg-Marquardt Method:** Incorporating the Levenberg-Marquardt method to the updating rule for $\mathbf{A}_k$, another modification has been also made to guarantee that the Hessian matrix **H** is positive defined, since this is a necessary condition for the matrix **H** to be invertible [1]. This modification is given by:

$$\Delta\mathbf{A}_k \propto p(C_k \mid \mathbf{x}_t, \theta)(\mathbf{H} + \mu\mathbf{I})^{-1}\mathbf{A}_k(\mathbf{I} - \mathbf{K}\tanh(\mathbf{s}_k)\mathbf{s}_k^T - \mathbf{s}_k\mathbf{s}_k^T)^T, \tag{29}$$

where μ is a small constant in (0,1).

In the experiments of this work, the EICAMM uses the Levenberg-Marquardt method in its learning rule for $\mathbf{A}_k$, as in Equation (29).

5 Experimental Results

To compare the performance of EICAMM to that presented by ICAMM, both approaches were used to classify random generated data and the well-known Iris data set. In the first case, random data having Laplace distributions was generated, varying the number of classes for $K = 2$ and $K = 3$ and the data dimension for $N = 2$ and $N = 3$. In these experiments, each class contains 1000 examples.

The iris flower data set [5] contains three classes with four numeric attributes of 50 instances each, where each class refers to a type of iris flower. One class is linearly separable from the other two, but the other two are not linearly separable from each other. The tests for this data set were performed for all the three classes in the data set and for only two classes that are linearly separable.

Tables 1 and 2 provide the random generated data classification results, in terms of overall accuracy, for EICAMM and ICAMM, respectively. It can be noted that EICAMM has significantly overperformed ICAMM in all cases, achieving its best performance for 2D artificial data with two classes.

The classification results, in terms of overall accuracy, for the Iris flower data set and both models are presented in Table 3. Here, EICAMM had also provided better results than those obtained by ICAMM.

Table 1. Classification Results for Random Generated Data – ICAMM

	2 classes	3 classes
2 dimensions	50.00%	33.00%
3 dimensions	50.00%	33.00%

Table 2. Classification Results for Random Generated Data – EICAMM

	2 classes	3 classes
2 dimensions	81.85%	76.65%
3 dimensions	79.70%	55.00%

Table 3. Classification Results for The Iris Flower Data Set

	ICAMM	EICAMM
2 Classes	54%	99%
3 Classes	25%	75.76%

6 Conclusions

This work introduced the Enhanced ICA Mixture Model (EICAMM), which incorporates some modifications to the original ICAMM algorithm learning rules, considering both modelling and implementation aspects. In order to evaluate the efficiency of the proposed model, a comparative study shows some results obtained by the EICAMM and the by the original one. It was noted that the proposed modifications can significantly improve the classification performance, considering random generated data and the well-known iris flower data set.

As future work, additional experimental tests, including image data for segmentation, will be performed.

Acknowledgement

The authors thank the Brazilian agency CAPES for the financial support.

References

1. Bazaraa, M. S.: Nonlinear Programming. John Wiley & Sons, Inc. (1979).
2. Bell, A. J., Sejnowski T. J. An information-maximization approach to blind separation and blind deconvolution. Neural Computation. Vol. 7 **6** (1995) 1129–1159.
3. Cardoso, J.-F., Laheld, B.: Equivariant Adaptive Source Separation. IEEE Transactions on Signal Processing. Vol. 45. **2** (1997) 434–444.
4. Comon, P.: Independent component analysis: A new concept ?. Signal Processing. Vol. 36. **3** (1994) 287–314.
5. Duda R., Hart P.: Pattern classification and scene analysis. Wiley. New York (1973).
6. Hyvärinen, A., Karhunen, J., Oja, E.: Independent component analysis. John Wiley & Sons (2001).
7. Jain A. K., Duin R. P. W., Mao J.: Statistical pattern recognition: A review. IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol. 22 **1** (2000) 4–37.
8. Lee T., Girolami M., Sejnowski T. J. Independent component analysis using an extended infomax algorithm for mixed sub-Gaussian and super-Gaussian sources. Neural Computation. Vol. 11 **2** (1999) 409–433.
9. Lee T., Lewicki M. S., Sejnowski T. J. ICA mixture models for unsupervised classification of non-Gaussian classes and automatic context switching in blind signal separation. IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol. 22 **10** (2000) 1078–1089.
10. Masters T.: Advanced algorithms for neural networks. Wiley. New York (1995).
11. De Ridder, D., Kittler, J., Duin, R.P.W.: Probabilistic PCA and ICA subspace mixture models for image segmentation, in: Mirmehdi, M. and Thomas, B. Proceedings British Machine Vision Conference 2000 (BMVC 2000, Bristol, UK), (2000) 112–121.
12. Shah C. A., Arora M. K., Robila S. A., Varshney P. K.: ICA mixture model based unsupervised classification of hyperspectral imagery. In Proceedings of 31st Applied Imagery Pattern Recognition Workshop (Washington, EUA), (2002) 112–121.

Appendix A - Derivation of Second Derivative for the Log-Likelihood Function

$$\Delta \mathbf{W} \propto [\mathbf{I} - \mathbf{K}\tanh(\mathbf{u})\mathbf{u}^T - \mathbf{u}\mathbf{u}^T]\mathbf{W} \tag{30}$$

Computing the second derivative in relation W of the equation (30) we have:

$$(\mathbf{I} - \mathbf{K}\tanh(\mathbf{u})\mathbf{u}^T - \mathbf{u}\mathbf{u}^T) + \frac{\partial(\mathbf{I} - \mathbf{K}\tanh(\mathbf{u})\mathbf{u}^T - \mathbf{u}\mathbf{u}^T)}{\partial \mathbf{W}}\mathbf{W} =$$

$$(\mathbf{I}-\mathbf{K}\tanh(\mathbf{u})\mathbf{u}^T-\mathbf{u}\mathbf{u}^T)+(-\mathbf{K}\tanh(\mathbf{u})\mathbf{s}^T\mathbf{A}^T-\mathbf{K}\mathrm{sech}^2(\mathbf{u})\mathbf{A}\mathbf{s}\mathbf{u}^T-\mathbf{u}\frac{\partial \mathbf{u}^T}{\partial \mathbf{W}}-\frac{\partial \mathbf{u}}{\partial \mathbf{W}}\mathbf{u}^T)\mathbf{W}=$$

$$(\mathbf{I}-\mathbf{K}\tanh(\mathbf{u})\mathbf{u}^T-\mathbf{u}\mathbf{u}^T)-\mathbf{K}\tanh(\mathbf{u})\mathbf{u}^T-\mathbf{K}\mathrm{sech}^2(\mathbf{u})\mathbf{u}\mathbf{u}^T-\mathbf{u}\mathbf{A}\mathbf{s}_t\mathbf{W}-\mathbf{A}\mathbf{s}_t\mathbf{u}^T\mathbf{W}$$

Using the result $\mathbf{u}_t = \mathbf{W}\mathbf{x}_t = \mathbf{W}\mathbf{A}\mathbf{s}_t$ and substituting in this expression, we have:

$$\mathbf{I} - \mathbf{K}\tanh(\mathbf{u})\mathbf{u}^T - \mathbf{u}\mathbf{u}^T - \mathbf{K}\tanh(\mathbf{u})\mathbf{u}^T - \mathbf{K}\mathit{sech}^2(\mathbf{u})\mathbf{u}\mathbf{u}^T - 3\mathbf{u}\mathbf{u}^T =$$

$$\mathbf{I} - 2\mathbf{K}\tanh(\mathbf{u})\mathbf{u}^T - \mathbf{K}\mathrm{sech}^2(\mathbf{u})\mathbf{u}\mathbf{u}^T - 3\mathbf{u}\mathbf{u}^T$$

Analysis of Galactic Spectra Using Active Instance-Based Learning and Domain Knowledge

Olac Fuentes[1], Thamar Solorio[1], Roberto Terlevich[1,2], and Elena Terlevich[1,2]

[1] Instituto Nacional de Astrofísica Óptica y Electrónica,
Luis Enrique Erro # 1,
Santa María Tonantzintla, Puebla, México 72840
{fuentes, thamy, rjt, eterlevi}@inaoep.mx
[2] Institute of Astronomy, Madingley Road,
Cambridge CB3 0HA, United Kingdom

Abstract. In this paper we present an efficient solution, based on active and instance-based machine learning, to the problem of analyzing galactic spectra, an important problem in modern cosmology. The input to the algorithm is the energy flux received from the galaxy; its expected output is the set of stellar populations and dust abundances that make up the galaxy. Our experiments show very accurate results using both noiseless and noisy spectra, and also that a further improvement in accuracy can be obtained when we incorporate prior knowledge obtained from human experts.

Keywords: active learning, instance-based learning, locally-weighted regression, prior knowledge.

1 Introduction

In astronomy and other scientific disciplines, we are currently facing a massive data overload. With the development of new automated telescopes dedicated to systematic sky surveys, archives that are several terabytes in size are being generated. For example, the Sloan Digital Sky Survey [1], currently underway, will provide astronomers with high-quality spectra of several million galaxies. A complete analysis of these spectra will undoubtedly provide knowledge and insight that can improve our understanding of the evolution of the universe. Such analysis is, however, impossible using traditional manual or semimanual means, thus automated tools will have to be developed.

In recent years, astronomers and machine learning researchers have started collaborating towards the goal of automating the task of analyzing astronomical data. For example, successful approaches for automated galaxy classification have been proposed using decision trees [2], neural networks [3, 4] and ensembles of classifiers [5, 6].

For problems that are inherently high-dimensional, such as analysis of galactic spectra, machine learning approaches have been less successful. In high-dimensional spaces, learning algorithms face the well-known curse of dimensionality [7], that states that the number of training examples needed to approximate a function accurately grows exponentially with the dimensionality of the task.

In this paper we propose a method that allows to approximate non-linear multidimensional functions using a small initial training set and active learning to augment this training set as needed, according to the elements of the test set. We apply this method to the problem of analysis of galactic spectra, in which one seeks to determine, from the spectrum of a galaxy, the individual stellar populations that make it up, as well as the abundance of dust in it. Our method allows to take advantage of prior domain knowledge, which is used to further increase the accuracy of the results obtained.

The organization of the remainder of this paper is as follows. Section 2 describes the problem of analyzing stellar spectra; Section 3 presents our method for function approximation, Section 4 presents experimental results and Section 5 presents conclusions and proposes directions for future work.

2 Analysis of Galactic Spectra

Nearly all information about a star is contained in its spectrum, which is a plot of flux against wavelength. By analyzing a galaxy spectrum we can derive valuable information about its star formation history, as well as other physical parameters such as its metal content, mass and shape. The accurate knowledge of these parameters is very important for cosmological studies and for the understanding of galaxy formation and evolution. Template fitting has been used to carry out estimates of the distribution of age and metallicity from spectral data. Although this technique achieves good results, it is very expensive in terms of computing time and therefore can be applied only to small samples.

2.1 Modelling Galactic Spectra

Theoretical studies have shown that a galactic spectrum can be modelled with good accuracy as a linear combination of three spectra, corresponding to young, medium and old stellar populations, together with a model of the effects of interstellar dust in these individual spectra.

Interstellar dust absorbs energy preferentially at short wavelengths, near the blue end of the visible spectrum, while its effects on longer wavelengths, near the red end of the spectrum, are small. This effect is called *reddening* in the astronomical literature.

Let $f(\lambda)$ be the energy flux emitted by a star or group of stars at wavelength λ. The flux detected by a measuring device is then $d(\lambda) = f(\lambda)(1 - e^{-r\lambda})$, where r is a constant that defines the amount of reddening in the observed spectrum and depends on the size and density of the dust particles in the interstellar medium.

A simulated galactic spectrum, $g(\lambda)$, can be built given c_1, c_2, c_3, the relative contributions of young, medium and old stellar populations, respectively; their reddening parameters r_1, r_2, r_3, and the ages of the populations a_1, a_2, a_3.

$$g(\lambda) = \sum_{i=1}^{3} c_i s(a_i, \lambda)(1 - e^{-r_i \lambda})$$

where $g(\lambda)$ is the energy flux detected at wavelength λ and $s(a_i, \lambda)$ is the flux emitted by a stellar population of age a_i at wavelength λ.

Therefore, the task of analyzing an observational galactic spectrum t consists of finding the values of $c_1, c_2, c_3, r_1, r_2, r_3, a_1, a_2$ and a_3 that minimize:

$$\sum_{\lambda} (t(\lambda) - g(\lambda))^2$$

Clearly, $c_1, ..., c_3$ have to be non-negative, and sum up to 1, also, realistic values of $r_1, ..., r_3$ are in the narrow range $[1 \times 10^{-5}, 6 \times 10^{-4}]$, and using only a few discrete values for a_1, a_2 and a_3 normally suffices for an accurate approximation. In particular, for our experiments we consider $a_1 \in \{3 \times 10^6\}$, $a_2 \in \{10^8, 3 \times 10^8, 5 \times 10^8, 8 \times 10^8\}$, and $a_3 \in \{10^9, 2 \times 10^9, 3 \times 10^9, 5 \times 10^9, 10^{10}\}$.

The next subsection describes the method we propose to solve this problem.

3 The Methods

In the problem we are trying to solve here, galactic spectral analysis, the algorithm tries to predict the reddening parameters for three age populations from a high dimensional spectrum. To circumvent the curse of dimensionality, we partition the problem into three subproblems, each of which is amenable to be solved by a different method. The key observation is that if we knew the values of the reddening parameters we could just perform a brute-force search over the possible combinations of values for the ages of stellar populations (a total of $1 \times 4 \times 5 = 20$) and for each combination of ages find the contributions that best fit the observation using least squares. Then the best overall fit would be the combination of ages and contributions that resulted in the best match to the test spectrum. Thus the crucial subproblem to be solved is that of determining the reddening parameters.

Predicting the reddening parameters from spectra is a difficult non-linear optimization problem, specially for the case of noisy spectra. We propose to solve it using an iterative active learning algorithm that learns the function from spectra to reddening parameters. In each iteration, the algorithm uses its training set to build an approximator to predict the reddening parameters of the spectra in the test set. Once the algorithm has predicted these parameters, it uses them to find the combination of ages and contributions that yield the best match to the observed spectra. From these parameters we can generate the corresponding spectrum, and compare it with the spectrum under analysis, if they are a close match, then the parameters found by the algorithm are correct, if not, we can add the newly generated training example (the predicted parameters and their corresponding spectrum) to the test set and proceed to a new iteration. Since this type of active learning adds to the training set examples that are progressively

Table 1. Pseudocode of our Active Learning Algorithm

```
0. Let $T$ be the test spectra
1. $S = \{\}$
2. For $i = 1$ to n
   a. Generate random parameter vector $p = [c_1, c_2, c_3, r_1, r_2, r_3, a_1, a_2, a_3]$
   b. Generate spectra $s$ according to $p$
   c. $S = S \cup \{\langle\langle s\rangle, \langle r_1, r_2, r_3\rangle\rangle\}$
3. While $T \neq \{\}$ do:
   a. Build $C$, an ensemble of approximators using LWLR
   b. For every test spectra $t \in T$
     - Use $C$ to predict the reddening parameters $r_1, r_2, r_3$ of $t$
     - For every triple $\langle a_1, a_2, a_3\rangle \in \{3 \times 10^6\} \times \{10^8, 3 \times 10^8, 5 \times 10^8, 8 \times 10^8\} \times$
       $\{10^9, 2 \times 10^9, 3 \times 10^9, 5 \times 10^9, 10^{10}\}$
       $R = \begin{bmatrix} s(a_1,\lambda_1)(1-e^{-r_1\lambda_1}),...,s(a_1,\lambda_m)(1-e^{-r_1\lambda_m}) \\ s(a_2,\lambda_1)(1-e^{-r_2\lambda_1}),...,s(a_2,\lambda_m)(1-e^{-r_2\lambda_m}) \\ s(a_3,\lambda_1)(1-e^{-r_3\lambda_1}),...,s(a_3,\lambda_m)(1-e^{-r_3\lambda_m}) \end{bmatrix}$
       $[c_1, c_2, c_3] = t(R^T R)^{-1} R^T$
       Generate spectra $g$ according to $q = [c_1, c_2, c_3, r_1, r_2, r_3, a_1, a_2, a_3]$
       $error(q) = \sum_\lambda (g(\lambda) - t(\lambda))^2$
     - Let $q^* = argmin\ error$
     - If $error(q^*) < threshold$
       output $\langle t, q^*\rangle$
       $T = T - \{t\}$
     - Else $S = S \cup \{\langle\langle s\rangle, \langle r_1, r_2, r_3\rangle\rangle\}$
```

closer to the points of interest, the errors are guaranteed to decrease in every iteration until convergence is attained.

An outline of the algorithm is given in Table 1. In steps 1 and 2, the algorithm constructs an initial training set, randomly generating parameter vectors (the target function) and computing their corresponding spectra (the attributes). Step 3 is repeated until a satisfactory fit has been found for every spectrum in the test set. First an approximator is built to derive reddening parameters from spectra using an ensemble and locally-weighted linear regression (LWLR), a well-known instance-based learning algorithm. Using this approximator, it tries to find the reddening parameters of each spectrum in the test set. Given the candidate reddening parameters found, it searches for the combination of ages and contributions that yield the minimum squared error, if the error is smaller than a threshold, it outputs the set of parameters found for that spectrum and removes it from the test set, if the error is not small enough, it adds the new training example to the training set and continues.

It should be pointed out that the active learning algorithm is independent of the choice of learning algorithm used to predict the reddening parameters. Any algorithm that is suitable to predict real-valued target functions from real-valued attributes could be used. In this work we use an ensemble of locally-weighted linear regression (LWLR). For building the ensemble we used a method that manipulates the attribute set. Here, each member of the ensemble uses a different subset randomly chosen from the attribute set. More information concerning ensemble methods, such as boosting and error-correcting output coding, can be found in [8]. We choose LWLR as the base learning algorithm because it has been applied successfully to several complex problems and, like all instance-based learning algorithms, it does not require any training time, thus it is very well suited to situations in which the training set is continually modified, as is the case with our method. For a description of LWLR we refer the reader to [9].

4 Experimental Results

In all the experiments reported here we use the following procedure: we generated randomly a set of galactic spectra with their corresponding parameters. This set was randomly divided into two equally sized disjoint subsets, one subset was used for training and the other was considered the test set. This procedure was repeated 10 times, and we report here the overall average.

In the first set of experiments our objective was to measure empirically the advantage of the active learning procedure versus a traditional ensemble of LWLR. As mentioned previously, the ensembles were constructed selecting randomly a subset of the attributes. In order to make an objective comparison, both methods used the same attribute subset and an ensemble of size 5. Table 2 shows mean absolute errors in the prediction of reddening and the relative contributions, also we present the root mean squared error between the predicted and the real test galactic spectra. We can see that a considerable error reduction is attained by our optimization algorithm in the estimation of reddening parameters, and that it yields a more accurate computation of the relative contribution of age populations.

4.1 Incorporating Prior Knowledge

Experimental results presented above show that the active learning procedure is a very accurate method in the determination of star formation history in galaxies. Those experiments used as attributes the complete energy flux from the galaxy. In contrast, an expert astronomer can derive, with very high accuracy, physical

Table 2. Comparison between an ensemble of LWLR (*LWLR-ensemble*) and active learning (*A*)

Algorithm	M.A.E. Contributions	M.A.E. Reddening	R.M.S.E. Flux
LWLR-ensemble	0.0821	2.0583e-004	8.4188e-007
A	0.0773	1.8030e-004	6.0688e-007

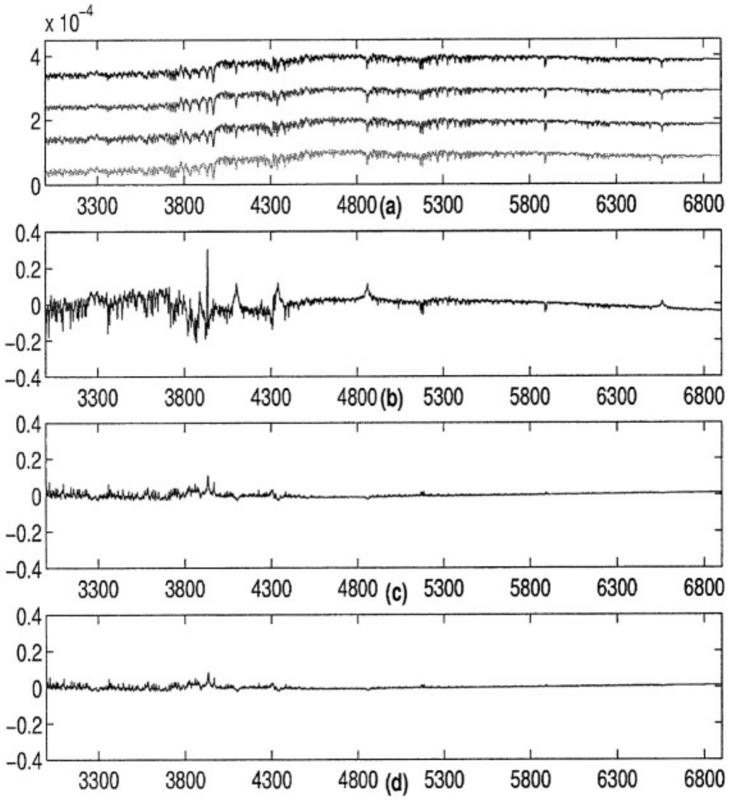


Fig. 1. Graphical comparison of results. Figure (a) from top to bottom and shifted by a constant to aid visualization: original test spectrum, spectrum recovered using ensemble of LWLR and original data, spectrum recovered using original data and active learning, and spectrum recovered using active learning and prior knowledge. Figures from (b) to (d) show relative difference between test spectrum and predicted spectra in the same listed order

parameters of a star by measuring only certain regions of the spectrum along the wavelength. These regions are known in the literature as Lick indices and were defined in 1973 by Faber et al. [10]. Although Lick indices were defined when model resolution was very low, astronomers have been using them for several decades now. We decided to explore in the following experiments whether the use of this prior knowledge can help machine learning algorithms to provide a more accurate prediction.

We introduce prior knowledge in the learning task by augmenting the relevance of the Lick indices when LWLR calculates the Euclidian distance from the query point to the training data. What we did was multiply the energy fluxes in the wavelengths corresponding to the Lick indices by a constant $k = 4$. That is, fluxes in regions defined by Lick indices were deemed to be 4 times as important as pixels in other regions. This value of k was set experimentally with a 10-fold cross-validation procedure. Table 3 presents prediction errors using prior knowledge. Figures for row *PK-LWLR* show results of using prior knowledge and an ensemble of LWLR, row *A+PK* presents results of the active learning algorithm and prior knowledge. If we compare figures from Table 2 and Table 3 we can see that even though the active learning algorithm does better than *PK-LWLR,* prior knowledge with active learning,*A+PK,* yields the best results. In Figure 1 we present a graphical comparison among using original data with an ensemble of LWLR, active learning using original data and prior knowledge with active learning. In figure (a) we present the original test spectrum and the reconstructed spectra from each of the three methods. Figures (b) to (d) show the relative difference in flux in predicted spectra presented in the same listed order. The relative difference in flux from the active learning algorithm that exploits prior knowledge is closer to a straight line, while the residuals from using the original data and the ensemble of LWLR has some high peaks indicating an error of nearly 25% in some spectral regions.

Table 3. Comparison between using prior knowledge with an ensemble of LWLR (*PK-LWLR*) and prior knowledge with active learning (*A+PK*)

Algorithm	M.A.E. Contributions	M.A.E. Reddening	R.M.S.E. Flux
PK-LWLR	0.0840	1.9251e-004	7.5726e-007
A+PK	0.0655	1.3189e-004	5.0766e-007

4.2 Noisy Data Experiments

Results presented above are encouraging. However, the data used in those experiments was noise free. We are aware that noisy data pose a more realistic evaluation of our algorithm, given that in real data analysis problems noise is always present. Galactic spectral analysis is no exception to this rule, noise can come from the source itself or from a bad calibration of the instruments. For this reason, we performed a new set of experiments aimed at exploring the noise-sensitivity of our active learning algorithm. We performed the same procedure described previously, except that this time we added to the test data a gaussian noise with zero mean, standard deviation of one and we use a ratio of signal to noise equal to 50.

Table 4. Comparison between an ensemble of LWLR, active learning and active learning with prior knowledge using noisy test data

Algorithm	M.A.E. Contributions	M.A.E. Reddening	R.M.S.E. Flux
LWLR ensemble	0.1145	2.6589e-004	1.1732e-006
A	0.1058	2.6533e-004	8.0577e-007
A+PK	0.0880	2.2244e-004	8.5577e-007

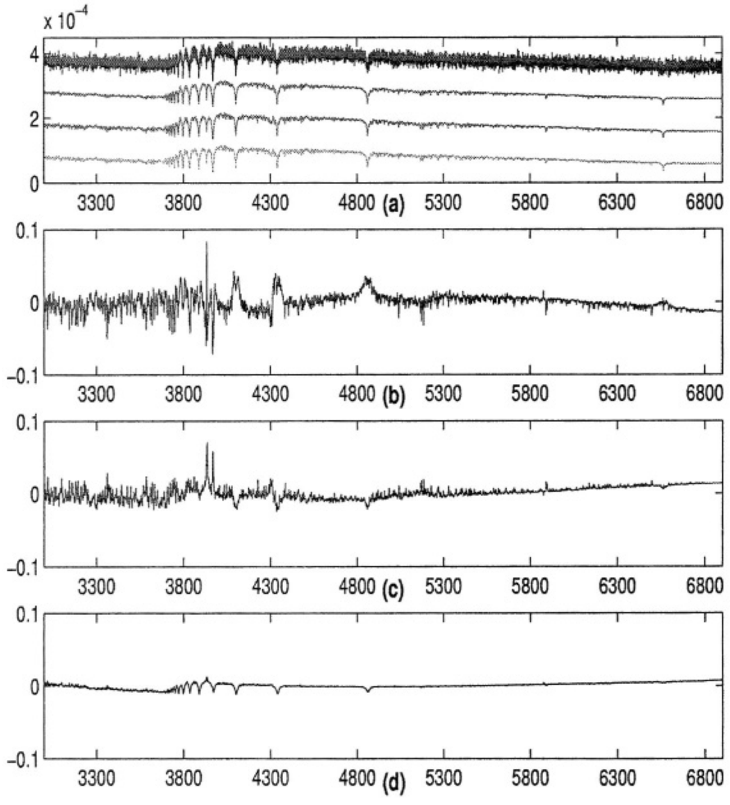


Fig. 2. Graphical comparison of results using noisy data. Figure (a) from top to bottom and shifted by a constant to aid visualization: original test spectrum, spectrum recovered using ensemble of LWLR and original data, spectrum recovered using original data and active learning, and spectrum recovered using active learning and prior knowledge. Figures from (b) to (d) show relative difference between test spectrum and predicted spectra in the same listed order

In order to deal with noisy data we used standard principal component analysis. Principal component analysis seeks a set of M orthogonal vectors v and their associated eigenvalues k which best describes the distribution of the data. This module takes as input the training set, and finds its principal components (PCs). The noisy test data are projected onto the space defined by the first 20 principal components, which were found to account for about 99% of the variance in the set, and the magnitudes of these projections are used as attributes for the algorithm. Table 4 shows results comparing, as before, results for an ensemble of LWLR using the original noisy data, active learning using original data and prior knowledge with active learning. As expected, when using noisy data the best results have higher error rates than the results presented in Table 3. However, when using prior knowledge and active learning *(A+PK)*, the prediction of reddening parameters is more accurate than with the other two methods: LWLR ensemble using the original data and active learning using original data *(A)*. Figure 2 presents a graphical comparison of results.

A relevant difference between noisy data experiments and noiseless ones is that the root mean squared error in the energy flux attained with active learning and the original data is lower than the one from using prior knowledge with active learning. It is somewhat surprising, given that in the other two error comparisons the opposite occurs. We consider that this may be due to the influence of giving a higher weight to the Lick indices when computing the reddening parameters, as well as the pseudoinverse. When the algorithm is given the original data it tries to minimize the root mean squared error along the complete energy flux, and each difference in λ along the spectrum has the same importance in the calculation of this error. This implies that it is not very important if in a few spectral lines the error is a little high, as long as for the vast majority the difference in each λ is minimum. In contrast, by using prior knowledge we are trying to minimize the error in a few spectral regions that are weighted more heavily, while the rest of the spectrum has a rather low weight in the calculation of the root mean squared error.

5 Conclusions

In this paper we have presented a learning algorithm for optimization that has a very strong feature: the ability of extending the training set automatically in order to best fit the target function for the test data. There is no need for manual intervention, and if new test instances need to be classified the algorithm will generate as many training examples as needed.

We have shown experimental results of the application of our method to solve the problem of, given a large set of galactic spectra, finding their corresponding star formation history and reddening. The method yields very accurate results, specially when using prior knowledge, even in the presence of noise.

Present and future work includes:

– Testing the method using real galactic spectra, taken from the Sloan Digital Sky Survey.
– Using models with different metallicities and explore the performance of our method when a new parameter, named metallicity, is introduced to the problem.
– Exploring a different method for introducing prior knowledge.
– Testing the active learning algorithm in other scientific data analysis tasks.

Acknowledgements

We would like to thank CONACYT for partially supporting this work under grant 166934 and Sandro Bressan and Miguel Cerviño for stimulating discussions.

References

1. A. Szalay and J. Gray. The World-Wide Telescope. *Science,* 293:2037–2040, September 2001.
2. E. A. Owens, R. E. Griffiths, and K. U. Ratnatunga. Using oblique decision trees for the morphological classification of galaxies. *Monthly Notices of the Royal Astronomical Society,* 281:153–157, 1996.
3. S. C. Odewahn, S. H. Cohen, R.A. Windhorst, and N. S. Philip. Automated galaxy morphology: A Fourier approach. *The Astrophysical Journal,* 568:539–557, 2002.
4. S. N. Goderya and S. M. Lolling. Morphological Classification of Galaxies using Computer Vision and Artificial Neural Networks: A Computational Scheme. *Astrophysics and Space Science,* 279:377–387, 2002.
5. D. Bazell and D. A. Aha. Ensembles of classifiers for morphological galaxy classification. *The Astrophysical Journal,* 548:219–223, 2001.
6. J. de la Calleja and O. Fuentes. Machine learning and image analysis for morphological galaxy classification. *Monthly Notices of the Royal Astronomical Society,* 349:87–93, March 2004.
7. R. E. Bellman. *Dynamic Programming.* Princeton University Press, Princeton, NJ, 1957.
8. T. Dietterich. Ensemble methods in machine learning. In *First International Workshop on Multiple Classifier Systems, Lecture Notes in Computer Science,* pages 1–15, New York: Springer Verlag, 2000. In J. Kittler and F. Roli (Ed.).
9. Christopher G. Atkeson, Andrew W. Moore, and Stefan Schaal. Locally weighted learning. *Artificial Intelligence Review,* 11:11–73, 1997.
10. S. M. Faber. Variations in Spectral-Energy Distributions and Absorption-Line Strengths among Elliptical Galaxies. *ApJ,* 179:731–754, February 1973.

Adapting Evolutionary Parameters by Dynamic Filtering for Operators Inheritance Strategy*

Xavier Bonnaire and María-Cristina Riff

Department of Computer Science, Universidad Técnica Federico Santa María, Valparaíso, Chile
{Xavier.Bonnaire, María-Cristina.Riff}@inf.utfsm.cl

Abstract. In this paper, we introduce a new idea to reduce the hard time consuming task of finding the set of initial parameter values of an evolutionary algorithm that uses the inheritance strategy. The key idea is to adapt the parameter values during the execution of the algorithm. This adaptation is guided by the degree of difficulty shown by the problem, which is being solved, for the algorithm. This strategy has been tested using an evolutionary algorithm to solve CSPs, but can easily be extended to any evolutionary algorithm which uses inheritance. A set of benchmarks showed that the new strategy helps the algorithm to solve more problems than the original approach.

1 Introduction

Designing genetic algorithms to tackle a complex problem is a time consuming task. It requires creating a new algorithm for each specific problem. Roughly speaking, we need to define at least the following:

– a representation,
– some especially designed operators
– the "best" set of parameter values or, an associated control strategy

Parameter control is better than tuning parameters to solve NP-hard and constrained problems, [7]. Many kinds of parameter control techniques have been proposed in the literature, however they are usually not cooperative ones. In this paper, we propose a way of increasing the cooperation between inheritance operators and filtering, using the problem complexity.

In the approach that uses inheritance, given a pool of operators available to be applied, the algorithm itself selects the "most appropriate" operator to be used. The offspring not only inherits the variable values from its parents, but it also inherits the operator which has been used to create one of them. Using inheritance from the parents behaviour, it is able to discard the operators which are not appropriate to solve a given problem. The goal is to propose a strategy that can help the evolutionary algorithm that uses inheritance to select a good set of operator

* Supported by Fondecyt Project 1040364.

parameter values. The inheritance algorithm itself is able to change its parameter values during the run depending on the state of the search. However, we know that the initial set of parameter values are relevant to obtain a better performance.

The tests presented in this paper use three well known crossover operators for inheritance GPX [9], Arc-crossover, Self-arc-crossover [13] especially designed for graph coloring problems.

The motivation of this work is to show that coupling a more refined mechanism to an evolutionary algorithm that uses inheritance, can improve its performance helping the algorithm to solve a greater number of problems.

The paper is organized as follows: In the next section we briefly describe the inheritance strategy. In section three we introduce the new parameter control strategy called "Dynamic Filtering Parameters Values" (DFPV). Section four presents the results obtained using both random generated graph coloring problems and specialized crossover operators. Finally, in the last section we present the conclusions and the future work.

2 Operators Inheritance Algorithm

In the previous research we used inheritance to find the most appropriate operator of a pool of operators that could potentially generate a better offspring. In this approach, the offspring not only inherits the variables values from its parents, but it also inherits the operator that has been used to create one of them. For instance, the inheritance process applied for crossover operators which needs two parents to create one child is shown in the figure 1.

Using this strategy we have concluded in [14] that:

– Given a pool of operators proposed in the literature in an evolutionary algorithm, it is able to select the best operator to be applied during the search.

```
Begin /* Procedure Operators Inheritance Algorithm */
Parent 1 = select(population(t))
Parent 2 = select(population(t))
if both parents have been generated by the same crossover operator
cross-op then
   Generate the offspring using cross-op
if only one parent have been generated by a crossover operator
cross-op then
   Generate the offspring using cross-op
if the parents have been created by different crossover operators
cross-op-1, cross-op-2 then
   Generate the offspring using the crossover operator used to
   create the best of the two parents.
Include the applied crossover operator information on the child
End /* procedure */
```

Fig. 1. Operators Inheritance Algorithm

– At the end of the algorithm, the worst operators have been discarded, and most of the time, the last population is composed of chromosomes generated by the best operators. Thus, the algorithm can identify which operators are not suitable for this problem during the search.

2.1 Specialized Crossover Operators for 3-Graph Coloring

Special attention is given to the constrained problems in the evolutionary community, [3] because they are especially hard for evolutionary algorithms. For the tests we have selected the 3-graph coloring problem which is an NP-hard constraint satisfaction problem. Evolutionary Algorithms for constrained problems can be divided into two classes: EAs using adaptive fitness functions and EAs using heuristics, [8], [12], [13]. We consider here three very well known heuristics based operators implemented in evolutionary algorithms for graph coloring

1. Greedy Partition Crossover Operator (GPX): GPX has been especially designed for the graph coloring problems and proposed in [9]. We use this operator because its results equal, and sometimes exceed those of the best well known algorithm for the graph coloring problems. A very important remark about GPX is that the edges of the graph, that is the constraints, are not involved in this crossover operator.
2. Arc-crossover Operator: We introduced in [13], [11] an EA for solving CSPs which uses information about the constraint network in the fitness function and in the genetic operators. The fitness function is based on the notion of *error evaluation* of a constraint. The error evaluation of a constraint is the number of variables of the constraint and the number of variables that are connected to these variables in the CSP graph.The crossover operator randomly selects two parents and builds a child by means of an iterative procedure over all the constraints of the CSP. Constraints are ordered according to their error-evaluation with respect to the instantiations of the variables that violate the constraints.
3. Constraint Dynamic Adapting Crossover: It uses the same idea of arc-crossover, that is there are no fixed points to make crossover. This operator uses a dynamic constraints priority. The dynamic constraint priority does not only take into account the network structure, but also the current values of the parents. The whole process is introduced, in detail, in [13].

2.2 Inheritance for 3-Graph Coloring Problems

Now, analysing in more detail the technique based on inheritance of the behaviour of the parents, we can observe that the parameter values of each operator changes during the search using the same initial and fixed probability. This probability is not changed in an explicit way. However, the crossover operator which has more probability to be chosen is the operator that has produced more individuals in the current population. For instance, if we decide to apply a crossover operator, the individual selection mechanism will choose two individuals from the current population. The operator that is more present in the population has

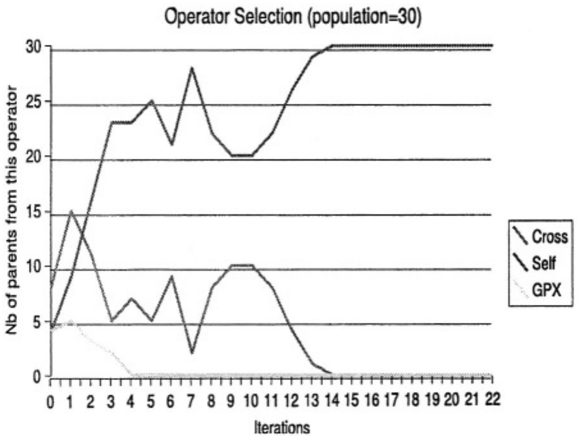


Fig. 2. Number of Operators Selection

more probability to be applied according to its fitness values. Thus, this technique can also be understood as a self-adaptive parameter control mechanism, according to the Michalewicz et al. classification in [7] This idea is illustrated in the figure 2. It shows that in the beginning the operators are used almost with the same frequency, but when the iteration number increases, the selection of the operators drastically reduces the use of some operators. In this case, GPX is no more useful after 10 iterations, that means that its probability to be applied after these 10 generations is zero, because there are no more individuals in the current population that have been created by GPX. Thus the algorithm has automatically removed this operator and continues its execution using the others ones.

3 Dynamic Filtering Parameters Values (DFPV)

The goal is to propose a strategy that can help the evolutionary algorithm which uses inheritance to select a good initial set of operators parameters values. The inheritance algorithm itself is able to change its parameter values during the execution depending on the state of the search. However, we have observed that the initial set of parameter values are relevant to obtain a better performance. Thus, the new strategy called Dynamic Filtering Parameters Values, DFPV, focuses on giving a good set of initial parameters values for the inheritance algorithm, thinking that it could change the parameters values during the search. Before introducing the strategy we need some preliminary definitions:

Definition 1. *(Set of Parameters Values)*
Given a set of j operators belonging to an evolutionary algorithm, we define a Set of Parameters Values S_p as the set $S_p = \{P_0, P_1, \dots, P_n\}$, such that each $P_i = \{p_{i1}, \dots, p_{ij}\}$, where p_{ij} is the probablity value of the jth operator in the P_i set.

Definition 2. *(Set of Problems Solved)*
Given a S_p, we identify a Set of Problems Solved by using the set P_i, $S_{solved}(P_i)$, as all the problems solved by the algorithm using these operators parameters values. We define the Set of Problems Solved by using the set S_p as the $S_{solved}(S_p) = Union\{S_{solved}(P_1), \ldots, S_{solved}(P_n)\}$.

Definition 3. *(Filter Set of Parameter Values)*
Given S_p and $S_{solved}(S_p)$, we define a Filter Set of Parameter Values, Q, where $Q = MINSET(S_p) = \{Q_1, \ldots, Q_k\}$ such that $S_{solved}(Q) = S_{solved}(S_p)$.

Remark 1. In the worst case $Q = S_p$, or in a equivalent way $k = n$.

In order to obtain the set $S_{solved}(S_p)$, the set of the problems that uses the parameters values in S_p allows the algorithm to find their solution. The algorithm runs for each P_i but for a small fixed number of iterations. The set Q represents the minimal set of S_p required to solve the same set of problems. It is possible that with both P_j and P_k the algorithm solves the same set of problems, thus only one of them is required to belong to Q. Finally, we define a hierarchy of the set Q whose elements belong to Q but they are ordered beginning with $H_1 = MAXSET(S_{solved}(Q_i))$, $\forall i = 1, \ldots, k$. That means, that the first in the hierarchy will be the set of the parameters values in Q which has solved most of the problems, and so on. Let *PR* be the initial set of problems, $PR = \{problem_1, ..., problem_x\}$.

Finally, the inheritance algorithm begins its execution using H_1 and continues with H_2, H_3 until H_k, in the worst case. We now use the previous hierarchy to try

```
Begin /* Procedure Construct Hierarchy */
E = PR
For each $P_i$ in $S_p$ do
    Execute for a small number of iterations Algorithm(E, $P_i$)
    E = E - $P_i$
endfor
$S_{solved}(S_p) = Union\{S_{solved}(P_1), \ldots, S_{solved}(P_n)\}$
Identify the Set $Q$ such that $S_{solved}(Q) = S_{solved}(S_p)$
$H = ordered(Q)$
```

Fig. 3. Procedure Construct Hierarchy

```
Begin /* Procedure DFPV */
E = PR
For h in H do
    Execute Algorithm(E, h)
    S = Union{ S, solved(h)}
    E = PR - S
    if E is empty then STOP
endfor
```

Fig. 4. Procedure DFPV

to solve the entire set of problems with a high number of iterations. Therefore, the number of problems we try to solve for each value decreases, or stays the same if none of them have been solved by the previous value.

Let *PR* be the initial set of problems, $PR = \{problem_1, ..., problem_x\}$ and S the set of already solved problems.

At the end of the hierarchy, some problems from the original set may not have been solved (E is not empty).

4 Tests

The goal of the following benchmarks is to evaluate the effect of including hierarchy to a dynamic operator selection into the constraints-graph based evolutionary algorithm, and to compare it with the original inheritance approach. The algorithm has been tested with randomly generated 3-coloring graphs, subject to the constraint that adjacent nodes must be colored differently. We used the Joe Culberson library, [6] to generate the random graphs. We have tested both algorithms for 3-coloring problems with solution, which are related to a sparse graph. These kinds of graphs are known to be the most difficult to solve [5]. For each number of constraints we have generated 500 random 3-coloring graph problems. In order to discard the "easy problems" we have applied DSATUR [2] to solve them. Then, we have selected the problems not solved by DSATUR. DSATUR is one of the best algorithms to solve this kind of problems. It is important to note that it is easy to find problems not solved by DSATUR in the hard zone [5], which is not the case with other connectivities.

4.1 Hardware

The hardware platform for the experiments was a PC Pentium IV, 1.4Ghz with 512 MB RAM under the LINUX operating system. The algorithm has been implemented in C.

4.2 Results

To run the preliminary step, we have chosen 200 iterations, and an initial set of parameters values for the crossover probability of {0, 0.1, ..., 1} used with our original inheritance algorithm.

We tried to solve each graph for 90, 120, 150, 180, 210, 240 and 360 constraints from the easiest one (90 constraints) to the hardest one (360 constraints). Note that this step, needed to compute Dynamic Filtering Parameter Value hierarchies only requires a few seconds to complete, which is not very time consuming. The number of 200 iterations is an arbitrary number chosen to be sufficient to be able to find the parameter values which give good results (in a number of solved problems). The preliminary step has found different hierarchies for each number of constraints. Then we run the same graphs with the Dynamic Filtering Parameter Value (DFPV) technique using the previously found hierarchies. To show the DFPV efficiency, we run the algorithm with the same number of 200

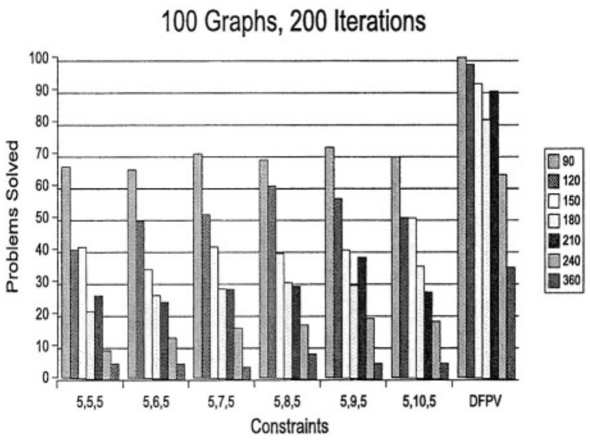


Fig. 5. Graph 200

iterations, which is rather few iterations to solve NP-Hard problems. The figure 5 shows the number of problems solved by the traditional algorithm compared to the number solved by the Dynamic Filtering Parameter Value technique. We can see that DFPV can find many more solutions than the traditional inheritance algorithm that uses the initial best parameter values for all the constraints. DFPV is able to solve 35% of the hardest problems with 360 constraints, which is, on average, 6 times better than the usual algorithm. Moreover, with a little number of iterations, the number of unsolved problems by the traditional algorithm linearily decreases when at the same time the DFPV maintains a significantly

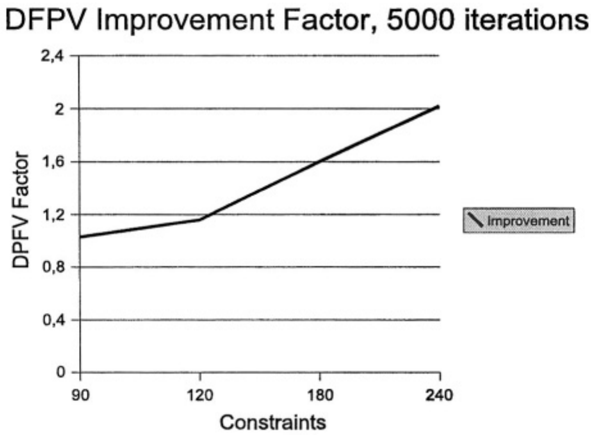


Fig.6. DFPV Factor

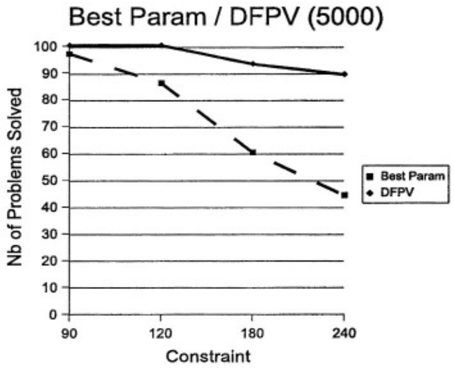


Fig. 7. Best DFPV

slower decreasing curve (especially between 90 and 210 constraints). This means that DFPV shows a very good efficiency for this constraints range.

The figure 6 gives the improvement factor of the DFPV with 5000 iterations with the hierarchy found using 200 iterations. A higher number of iterations really helps the traditional algorithm to solve more problems. Therefore, the difference in terms of solved problems between the two algorithms becomes smaller. But, the harder the problem is (the higher the number of constraints), the more efficient the DFPV is. We see that for 240 constraints, the DFPV solves twice as many more problems than the best parameter value for the traditional inheritance approach. The figure 7 shows the number of problems solved by both algorithms. It is clear that the delta between them quickly increases when the number of constraints becomes higher (harder problems).

In the figure 8, we can see that the traditional inheritance algorithm benefits from a higher iterations number. Its performance can be up to 900% better for 240 constraints with 5000 iterations than with 200 iterations. On the other

	Traditional Algorithm				DFPV			
	Iterations		Improvement		Iterations		Improvement	
Const	200	5000	Δ	%	200	5000	Δ	%
90	72	97	+25	34.7	100	100	0	0
120	56	86	+30	53.5	97	100	+3	3
180	28	60	+32	114	82	93	+11	13
240	5	45	+40	900	64	89	+39	39

Fig. 8. Results Comparison

hand, DFPV does not benefits as much as the non hierarchical approach of a high number of iterations. For 240 constraints, the gain is only 39% which is really much smaller. DFPV is able to solve 60% more of problems with 200 iterations than the traditional algorithm can do with 5000 iterations. Thus, to obtain much better results, in terms of solved problems, DFPV only requires hundreds of iterations, where the traditional approach needs thousands of iterations.

5 Conclusion

Our new approach with the Dynamic Filtering Parameter Value technique has shown very good results in terms of the number of problems solved, compared to the traditional inheritance method which begins with the best single value for a given parameter, previously determined by tuning. Coupled to the inheritance mechanism, the DFPV becomes more and more efficient when the problem becomes harder to solve (up to twice better for our 3-coloring graph problem with 240 constraints). The preliminary step required to find the parameter hierarchy which is going to be used for a given problem type is a very low time consuming task, as it only needs to run about 200 to 300 iterations. Moreover, the inheritance mechanism significantly reduces the number of iterations to find a solution for the hardest problems, as useless operators can often be guessed very quickly. DFPV also requires much less iterations than the original inheritance algorithm to solve a given problem than the traditional algorithm, and it gives much better results. As a consequence, DFPV requires less CPU time to find better results, which is very welcomed to solve large NP-Hard problems.

5.1 Future Work

For a given type of problem, it seems to be very interesting to be able to automatically find the smallest number of iterations required to find the best hierarchy to be used by DFPV. For example, 150 iterations would have produced a hierarchy as good as 200 iterations have done? Or just a few iterations more (210, 220,...) would have dramatically improved the number of solved problems? It would also be very interesting to have a parallel version of DFPV, as much of the code can be parallelized to further reduce the CPU time needed to solve the problems. We should also investigate the relation between the DFPV and the inheritance, to be able to have more cooperation between the two mechanisms. This could allow having a different order in the hierarchy used in DFPV to reduce the time required to find a maximum number of solutions.

References

[1] Bonnaire X., Riff M.-C, A Self-Adaptable Distributed Evolutionary Algorithm to Tackle Space Planning Problems. Applied Parallel Computing, Eds. J. Fagerholm, J. Hastaja, J. Järvinen, M. Lyly, P. Raback and V. Savolainen, LNCS 2367, ISSN 0302-9743, Springer-Verlag, pp. 403-410, 2002.

[2] Brelaz, New methods to color vertices of a graph. Communications of the ACM, 22,pp. 251-256, 1979.
[3] C. Coello, Theoretical and Numerical Constraint Handling Techniques used with Evoltuionary Algorithms. A Survey of the State of the Art. Computer Methods in Applied Mechanisc and Engineering, 191(11-12), pp. 1245-1287, 2002.
[4] B. Craenen, A. Eiben, E. Marchiori. Solving Constraint Satisfaction Problems with Heuristic-based Evolutionary Algorithms. In Proceedings of the Congress on Evolutionary Computation, (CEC2000),pages 1571-1577, IEEE Press.
[5] Cheeseman P.,Kanefsky B., Taylor W., Where the Really Hard Problems Are. Proceedings of IJCAI-91, pp. 163-169, 1991
[6] Culberson, J. http://web.cs.ualberta.ca/ joe/.
[7] Eiben A.E., Hinterding R., and Michalewicz Z. Parameter Control. Evolutionary Computation 2: Advanced Algorithms and Operators, pages 170-187, Institute of Physics Publishing, 2000.
[8] A.E. Eiben, J.I. van Hemert, E. Marchiori, A.G. Steenbeek. Solving Binary Constraint Satisfaction Problems using Evolutionary Algorithms with an Adaptive Fitness Function. Fifth International Conference on Parallel Problem Solving from Nature (PPSN-V), LNCS 1498, pp. 196-205, 1998.
[9] Hamiez J-F, Hao J., Scatter Search for Graph Coloring, Lecture Notes in Computer Science 2310: 168-179, 2002.
[10] J. Paredis, Coevolutionary Algorithms, The Handbook of Evolutionary Computation, 1st supplement, BSck, T., Fogel, D., Michalewicz, Z. (eds.), Oxford University Press.
[11] Riff M.-C., From Quasi-solutions to Solution: An Evolutionary Algorithm to Solve CSP. Constraint Processing (CP96), Ed. Eugene Freuder, pp. 367-381, 1996.
[12] Riff M.-C., Evolutionary Search guided by the Constraint Network to solve CSP. Proc. of the Fourth IEEE Conf on Evolutionary Computation, Indianopolis, pp. 337-342, 1997.
[13] Riff M.-C., A network-based adaptive evolutionary algorithm for CSP, In the book "Metaheuristics: Advances and Trends in Local Search Paradigms for Optimisation", Kluwer Academic Publisher, Chapter 22, pp. 325-339, 1998.
[14] Riff M.-C, Bonnaire X., Inheriting Parents Operators: A New Dynamic Strategy for Improving Evolutionary Algorithms. Foundations of Intelligent Sytems, Eds. M. Hacid, Z. Ras, D. Zighed and Y. Kodratoff, LNCS/LNAI 2366, ISSN 0302-9743, Springer-Verlag, pp. 333-341, 2002.

Collaborative Filtering Based on Modal Symbolic User Profiles: Knowing You in the First Meeting

Byron Bezerra, Francisco Carvalho, and Gustavo Alves

Centro de Informatica - CIn / UFPE,
Av. Prof. Luiz Freire, s/n - Cidade Universitaria,
CEP 52011-030 Recife - PE, Brazil
{bldb, fatc, gsa}@cin.ufpe.br

Abstract. Recommender systems seek to furnish personalized suggestions automatically based on user preferences. These systems use information filtering techniques to recommend new items which has been classified according to one of the three approaches: *Content Based Filtering, Collaborative Filtering* or *hybrid filtering methods.* This paper presents a new hybrid filtering approach getting the better qualities of the *kNN Collaborative Filtering* method with the content filtering one based on *Modal Symbolic Data.* The main idea is comparing modal symbolic descriptions of users profiles in order to compute the neighborhood of some user in the *Collaborative Filtering* algorithm. This new approach outperforms, concerning the *Find Good Items* task measured by *half-life utility* metric, other three systems: content filtering based on *Modal Symbolic Data, kNN Collaborative Filtering based on Pearson Correlation* and hybrid *Content-Boosted Collaborative* approach.

1 Introduction

Recommender systems allow E-commerce websites to suggest products to their costumers, providing relevant information to help them in shopping tasks. Additionally, most often these sort of systems have increasing their importance in entertainment domains [12]. For instance, some interesting features are personalized TV guides in digital televisions and music recommendation in on-line stations.

Independently of the domain, two recommendation tasks have been mainly used by information systems in order to minimize information overload problems to their users [7,12]. The first one is the presentation of a list of items ranked according its relevance to an active user. The second task is the item relevance estimation based on active user needs. According to [7], the former, known by *Fing Good Items,* is the core recommendation task recurring in a wide variety of commercial and research systems. Furthermore, in commercial system the score of each item presented in a recommendation list is not usually shown. Observe that a small score for some item may inhibit user to buy it.

Whatever is the recommender task, in order to execute it, recommender systems must acquire as much information as possible about user preferences. This information is generally stored in some machine understandable format called user profile. A relevant problem remains in this process: the user has not enough time to giving information about him/her. So, it is necessary to learn user preferences with a piece of information provided by him/her, possibly in the first system usage, and improving this learning in future user interactions.

The next step is filtering in relevant information in order to present it to the user through his/her profile previously acquired. The proposed solutions for this subject can be classified in two main groups concerning the kind of filtering approach, e. g., Content Based *(CB)* Filtering (which is based on the correlation between the user profile and items content) or Collaborative Filtering (which is based on the users profiles correlation). These techniques have inherent limitations, such as impossibility to codify some information in the first approach [1] and latency (or *cold-start* problem) in the second one [4,9,11]. Therefore, several works [1,4,9,10,11] have exploiting hybrid recommenders to overcome the drawbacks of each.

In this paper we present a new hybrid filtering approach mixing the better qualities of the kNN Collaborative Filtering *(kNN-CF)* [6] with the content filtering based on Modal Symbolic *(MS)* Data [5]. In our method the user profile is modeled by *MS* descriptions derived from the content of items previously evaluated by the user. This becomes very useful to compute the neighborhood of some active user, mainly in the beginning of user interactions with the system when there is little information about him/her. We show our method requires less information about the user to provide better accuracy recommendation lists as those generated with some algorithms.

This new approach is evaluated by comparing it with the pure content filtering based on *MS* Data [5], with the pure *kNN-CF* based on Pearson correlation [6] and with the Content Boosted Collaborative Filtering *(CBCF)* hybrid method described in [9]. This comparison is performed in the movie recommendation domain, where the user profile is formed by way of a list of items that the user either preferred or disliked in the past, along with their respective grades.

The description of each item is a row of a data table the columns of which are classical or symbolic variables. Each cell of this data table may have a single category (e.g., the *director* and *year* attributes) or a set of categories (e.g., the *genre* and *cast* attributes). Table 1 shows an example of an item (movie) description.

Table 1. Content description of a movie evaluated by some user

Variable	Variable type	Description
Director	Single valued qualitative nominal	Steven Spielberg
Cast	Multi-valued qualitative nominal	{Tom Hanks, David Morse}
...	...	...
Grade	Single valued qualitative ordinal	5

2 Related Work

The majority of the works related to hybrid information filtering *(IF)* methods may be classified in one of the following three categories, where each category has a particular strategy to combine a collaborative algorithm with a *CB* one.

The first approach is to weight the predictions computed by both *IF* algorithms with a suitable function in order to compute a final prediction. For instance, this idea has been firstly proposed by [4]. Also, in [10] it is proposed a unified probabilistic framework for merging collaborative and *CB* recommendations.

Another strategy is estimate the score of some unknown items in the repository for a subset of users through a *CB* algorithm and following this step, use a Collaborative Filtering *(CF)* algorithm to predict new items to some active user. The *GroupLens* research system [11] deployed some automated agents who simulate human behavior and calculate scores for new items in repository diminishing the *cold-start* problem. Additionally, in [9] the *CB* algorithm is used to convert a sparse ratings matrix into a full ratings matrix; and then uses *CF* to provide recommendations. So, this technique minimizes the sparsity problem and achieves good results in some contexts.

The third category is characterized by algorithms that build and maintain a user profile based on the content description of items previously evaluated by the user. These content profiles allow measuring the correlation between users, which is important to neighborhood definition in collaborative recommendations. For instance, in *Fab* [1] the user profile is based on the content of web pages previously evaluated.

In the next section, we present a new hybrid method classified in this last category. The main idea is to build and maintain the user profile with *MS* data using techniques such as defined in [5], which proposes a *CB* filtering method based on Symbolic Data Analysis (SDA) field. *SDA* provides suitable tools for managing aggregated data described by multi-valued variables, where data table entries are sets of categories, intervals, or probability distributions (for more details about *SDA* see the site http://www.jsda.unina2.it/).

3 Collaborative Filtering Based on Modal Symbolic User Profiles

The following steps are executed to generate recommendation lists in the *CF* algorithm based on *MS* user profiles:

1. *Construction of the modal symbolic descriptions of the user profile.* This step can be done incrementally without degrading the memory usage.
2. *Weight all users based on their similarity with the active user.* Similarity between users is measured by a suitable function which compares the *MS* descriptions of each user profile.
3. *Select the k closest users as neighbors of active user.* The closeness is defined by similarity between some candidate neighbor and the active user.
4. *Generation of a ranked list of items after computing predictions from a weighted combination of the selected neighbors' ratings.*

Although, the steps 2-4 are standard in *CF* algorithms, the **2nd** one is done in a *CB* way through the *MS* user profiles built in **1st** step. Before detailing all phases of our algorithm we need to introduce some issues related to SDA field.

3.1 Symbolic Data

Data analysis has a data table as input where the rows are the item descriptions and the columns are the variables. A single cell of this data table contains a single quantitative or categorical value. Often in the real world information is too complex for common data-type to describe. This is why different kinds of symbolic variables and symbolic data have been introduced [2]. For example, a categorical multi-valued variable for an item takes a subset of its domain as value.

In this paper we are concerned with *MS* data. Formally, let D_j be a finite set of categories. A modal variable y_j with domain D_j defined in the set $E = \{a, b,\dots\}$of objects is a multi-state variable where, for each object $a \in E$, not only is a subset of its domain D_j given, but also for each category m of this subset, a weight $w(m)$ is given that indicates how relevant m is for a. Formally, $y_j(a) = (S_j(a), q_j(a))$ where $q_j(a)$ is a weight distribution defined in $S_j(a) \subseteq D_j$ such that a weight $w(m)$ corresponds to each category $m \in S_j(a)$. $S_j(a)$ is the support of the measure $q_j(a)$ in the domain D_j. In our approach, a symbolic description of an item is a vector where there is a weight distribution in each component given by a *MS* variable.

In order to exemplify the interest of these kind of data, let us consider the movie domain. When recording movies information concerning a user profile g, we come across an example where the variable "Cast" (represented here as y_{Cast}) allows no single-valued answer, since the user has seen many films in the past. Therefore, it would be appropriate to record this variable as a *MS* variable:

$$y_{Cast}(g) = (\{\text{Tom Hanks, David Morse, Demi Moore}\}, \{0.30, 0.20, 0.50\})$$

which reads: with frequency of 30% Tom Hanks, with 20% David Morse, with 50% Demi Moore. In this example, $S_{Cast}(g) = \{\text{Tom Hanks, David Morse, Demi Moore}\}$ is the support of the measure $q_{Cast}(g) = \{0.30, 0.20, 0.50\}$.

3.2 Building the Modal Symbolic User Profile

The subject of this step is constructing a suitable representation of the user profile following the algorithm described in [5] with a few adaptations. So, in [5] it was proposed each user profile must be represented by a set of *MS* descriptions which synthesize the whole information given by the item descriptions belonging to the user profile, taking into account the user evaluation (grade) of each item of his profile.

The construction of the *MS* descriptions of the user profile involves two steps: pre-processing and generalization. The general idea is to build a *MS* description for each item evaluated by the user (pre-processing) and then aggregate these *MS* descriptions in some *MS* descriptions where each one represents a particular user interest (generalization).

Pre-processing. This step aims to associate with each item a *MS* description. It is necessary for both constructing the set of *MS* descriptions used to represent the user profile and comparing the user profile with a new item (in *CB* filtering) or with another user profile (important to step 2 of our recommendation algorithm).

Let $x_i = (X_i^1, \ldots, X_i^p, C(i))$ be the description of an item i (i=1, ..., n), where $X_i^j \subseteq D_j$ (j=1, ..., p) is a subset of categories of the domain D_j of the variable y_j and $C(i) \in D = \{1, \ldots, 5\}$ indicates the user evaluation (grade) for this item. For each category $m \in X_i^j$, we can associate the following weight:

$$w(m) = \frac{1}{\left|X_i^j\right|} \tag{1}$$

where $|X_j|$ is the number of elements belonging to X_j (its cardinality). Then, the *MS* description of item i is $\tilde{x}_i = (\tilde{X}_i^1, \ldots, \tilde{X}_i^p, C(i))$, where $\tilde{X}_i^j = \tilde{X}_j(i) = (S_j(i), q_j(i))$ and $\tilde{X}_j$ is a *MS* variable. $S_j(i) = X_i^j$ is the support of the weighted distribution $q_j(i)$.

The *MS* description of the item of Table 1 is displayed in Table 2. In this example, $S_{Cast}(i)$ = {Tom Hanks, David Morse} and $q_{Cast}(i)$ = {0.5, 0.5}. Notice that *grade* variable is not pre-processed. It is used to evaluate the item, not describe it.

Table 2. Content description of a movie evaluated by some user

Variable	Description
Director	({Steven Spielberg}, {1.0})
Cast	({Tom Hanks, David Morse}, {0.5,0.5})
...	...
Grade	5

Generalization. After pre-processing the items evaluated by the user, we are able to construct the symbolic descriptions of his/her profile. In our approach, each user profile is formed by a set of sub-profiles. Each sub-profile is modeled by a *MS* description that summarizes the entire body of information taken from the set of items the user has evaluated with the same grade.

Formally, let u_g be the sub-profile of user u which is formed by the set of items which have been evaluated with grade g. Let $y_{u_g} = (Y_{u_g}^1, \ldots, Y_{u_g}^p)$ be the *MS* description of the sub-profile u_g, where $Y_{u_g}^j = (S_j(u_g), q_j(u_g))$, with $S_j(u_g)$ being the support of the weighted distribution $q_j(u_g), j = 1, \ldots, p$.

If $\tilde{x}_i = (\tilde{X}_i^1, \ldots, \tilde{X}_i^p, C(i))$, where $\tilde{X}_i^j = (S_j(i), q_j(i))$ (j=1,...,p), is the MS description of the item i belonging to u_g, the support $S_j(u_g)$ of $q_j(u_g)$ is defined as

$$S_j(u_g) = \bigcup_{i \in u_g} S_j(i) \tag{2}$$

Let $m \in S_j(u_g)$ be a category belonging to D_j. Then, the weight $W(m) \in q_j(u_g)$ of the category m is defined as

$$W(m) = \frac{1}{\left|u_g\right|} \sum_{i \in u_g} \delta(i,m) \quad \therefore \quad \delta(i,m) = \begin{cases} w(m) \in q_j(i), & \textit{if } m \in S_j(i) \\ 0, & \textit{otherwise} \end{cases} \tag{3}$$

where $|u_g|$ is the number of elements belonging to the set u_g.

3.3 Comparing Modal Symbolic Profiles

In this step we describe a suitable function that measures the similarity between each *MS* description of user profiles. The hypothesis behind this function is that two users have similar preferences if each pair of their sub-profiles evaluated with same grade are similar. We use this function to define the neighborhood of some active user.

Let $y_{u_g} = (Y^1_{u_g}, \ldots, Y^p_{u_g})$ be the *MS* description of the sub-profile u_g of some active user. Also, let $y_{v_g} = (Y^1_{v_g}, \ldots, Y^p_{v_g})$ be the *MS* description of the sub-profile v_g of some candidate neighbor tor active user. The comparison between the active user u and the candidate neighbor v is achieved by the following similarity function:

$$\psi(u,v) = \frac{\sum_{g\in\{1,2,3,4,5\}} \left(1 - \phi(y_{u_g}, y_{v_g})\right)}{5} \tag{4}$$

where the function $\phi(y_{u_g}, y_{v_g})$, $g \in \{1,2,3,4,5\}$, has two components: a context free component, which compares the sets $S_j(u_g)$ and $S_j(v_g)$; and a context depend component, which compares the weight distributions $q_j(u_g)$ and $q_j(v_g)$. The dissimilarity function ϕ compares two *MS* descriptions variable-wise first by taking position and content differences into account. Next, it aggregates the partial comparison results. Other dissimilarity functions suitable to the comparison of *MS* descriptions are presented in [2], but none of them take into account differences in position that arise when the feature type is ordered. This function is defined as:

$$\phi(y_{u_g}, y_{v_g}) = \frac{1}{p}\sum_{j=1}^{p}\left[\phi_{cf}(S_j(u_g), S_j(v_g)) + \phi_{cd}(q_j(u_g), q_j(v_g))\right] \tag{5}$$

where ϕ_{cf} measures the difference in position in cases where sets $S_j(u_g)$ and $S_j(v_g)$ are ordered; and ϕ_{cd} measures the difference in content between y_{u_g} and y_{v_g}.

Table 3 expresses the agreement (α and β) and disagreement (γ and δ) between the weight distributions $q_j(u_g)$ and $q_j(v_g)$.

Table 3. Comparison between the weight distributions $q_j(u_g)$ and $q_j(v_g)$

		User u_g	
		+ (Agreement)	- (Disagreement)
User v_g	+	$\alpha = \sum_{m\in S_j(u_g)\cap S_j(v_g)} w(m)$ • $\beta = \sum_{m\in S_j(u_g)\cap S_j(v_g)} W(m)$	$\gamma = \sum_{m\in \overline{S_j(u_g)}\cap S_j(v_g)} w(m)$
	-	$\delta = \sum_{m\in S_j(u_g)\cap \overline{S_j(v_g)}} w(m)$	

The context dependent component ϕ_{cd} is defined as:

$$\phi_{cd}(q_j(u_g), q_j(v_g)) = \frac{1}{2}\left(\frac{\gamma+\delta}{\alpha+\gamma+\delta} + \frac{\gamma+\delta}{\beta+\gamma+\delta}\right) \tag{6}$$

If the domain D_j of the categorical variable y_j is ordered, let $m_L = min(S_j(u_g))$, $m_U = max(S_j(u_g))$, $c_L = min(S_j(v_g))$ and $c_U = max(S_j(v_g))$. The join [8] $S_j(u_g) \oplus S_j(v_g)$ is defined as:

$$S_j(u_g) \oplus S_j(v_g) = \begin{cases} S_j(u_g) \bigcup S_j(v_g), & \text{if the domain } D_j \text{ is non ordered} \\ \{\min(m_L, c_L), \max(m_U, c_U)\}, & \text{otherwise} \end{cases} \tag{7}$$

The context dependent component ϕ_{cf} is defined as:

$$\phi_{cf}(S_j(u_g), S_j(v_g)) = \begin{cases} 0, \ \text{if } S_j(u_g) \bigcap S_j(v_g) \neq \varnothing \\ \dfrac{|S_j(u_g) \oplus S_j(v_g)| - |S_j(u_g)| - |S_j(v_g)|}{|S_j(u_g) \oplus S_j(v_g)|}, & \text{otherwise} \end{cases} \tag{8}$$

Now that we are able to compute the similarity between the active user u with each user in the database, we can do the 3rd step in a straightforward manner.

3.4 Generating a Ranked List of Items

The subject of this step is generating a ranked list of items according to user needs. In order to achieve this goal we need to compute predictions for each unknown item in the repository using the neighborhood found in step 3. Therefore, consider the following function ρ that measures the relevance of some item i to some active user u:

$$\rho(u,i) = \bar{r}_u + \frac{\sum_{v=1}^{k} (r_{v,i} - \bar{r}_v) * \psi(u,v)}{\sum_{v=1}^{k} \psi(u,v)} \tag{9}$$

where k is the neighborhood size. At this point, we can sort the list of items according the values produced by equation 9 and present this ranked list to the user.

4 Experimental Evaluation

As described in section 1, our case study is movie recommendation domain. We use the Movielens[1] dataset joined with a content database crawled from IMDB[2] to perform experimental tests. This prepared dataset contains 91190 ratings in the interval of 1 (the worst grade) to 5 (the best grade) of 943 different users for 1466 movies. In order to setup our environment we must answer three questions:

1. *How much the system knows about some user to provide recommendation?*
2. *What is the recommendation task we would like to evaluate?*
3. *How can we evaluate this task and compare different system performances?*

[1] http://movielens.umn.edu
[2] http://www.imdb.com

Concerning the first question, in this paper we are interesting in evaluate the utility of a recommender system in the beginning of user interactions. In this scenario, the user has not provided a lot of information about himself/herself, first because there was no much time to do this, and second because it is not a good strategy of information systems ask lots of questions to the user whereas he/she would leave the system and never come back. Therefore, we think it is reasonably user evaluate about 5 or even 10 items in the first contact.

About the second question, our system is asked to furnish some ordered lists (*Find Good Items* task). This decision was motivated by the hypothesis that in an E-commerce environment this task is more useful than other available tasks of recommender systems [7, 12].

Concerning our third question above, according to [7], an adequate rank accuracy metric was proposed by [3] called *half-life utility* metric. This evaluation metric was specially designed for tasks such as *Find Good Items*. The most important advantage of such metric is it measures the utility of a sorted list taking into account the user generally observes the first results of this list. Then, it assumes the hypothesis that each successive item on a list is less likely to be viewed by the user according to an exponential decay. Fundamentally, the utility of a sorted list for a user is based on the probability of the item having been seen by the user, multiplied by the item utility itself. See [3, 7] for additional details of *half-life utility* metric.

At now we can describe the methodology used to perform our experiments. First, we select all users that have been evaluated at least 100 items of 1466 movies available. This users were used in test set to perform four different experiments concerning the number m={5,10} of items provided in training set for each user and the neighborhood's size k={30,50} used in collaborative and hybrid filtering algorithms. Additionally, it was running an adapted version of the standard 10 fold cross-validation stratified methodology [13 (pages 125:127)]. This adaptation consist in arrange the training set and test set, respectively, in the proportion of 1 to 9, instead of 9 to 1 as done in the standard schema. This will be compatible with the fact that the user does not furnish enough information in his/her first contact with the system. The following algorithms were executed in our tests:

1. (MSA) – Content-Based Information Filtering based on *MS* Data;
2. (CFA) – *kNN-CF* based on Pearson Correlation;
3. (CBCF) – Content-Boosted Collaborative Filtering using as *CB* predictor the MSA;
4. (CMSA) - Collaborative Filtering based on Modal Symbolic User Profiles.

In Table 4 we can see the average (μ) and standard deviations (σ) of *half-life utility* metric for all algorithms grouped by k={30,50} and m={5,10}. Also, this table presents the observed t-statistic values [13 (pages 129:132)] concerning the two-tailed test based on 9 degrees of freedom between average behavior of the CMSA$_{k=50,m=5}$ method and the average behavior of the others methods (MSA, CFA and CBCF).

Table 4. Results of experiments grouped by k (size of neighborhood) and m (number of items in user profile) according to *half-life utility* metric

		MSA			CFA			CBCF			CMSA	
k	m	μ	σ	t	μ	σ	t	μ	σ	t	μ	σ
30	5	34,34	0,78	47,28	40,21	4,32	11,91	20,51	7,30	16,16	44,67	11,87
	10	31,84	1,11	65,70	58,09	1,99	6,32	42,14	10,22	6,54	53,49	9,71
50	5	34,34	0,78	47,28	40,21	4,32	11,91	20,51	7,30	16,16	**63,87**	2,21
	10	31,84	1,11	65,70	58,09	1,99	6,32	42,14	10,22	6,54	63,57	2,00

As we hope, the MSA algorithm has the same performance independently of *k* value. As well, we can see this happens to CFA and CBCF algorithms. In fact, as pointed out by [6] the quality of recommendations performed by *CF* systems does not improve significantly when the number of neighbors is higher than 30.

On the other hand, the value of 50 for *k* is determinant to improve the quality in CMSA system. Moreover, it arises that having 50 neighbors and just 5 items in user profile is sufficient to CSMA achieve the best recommendation lists than each other algorithm considered in our analysis (with a confidence level of 0.1%). It is important to say that in real systems it is not a problem reach 50 neighbors since this systems easily keep thousands of users. Nevertheless, having good recommendations with just 5 items can help systems accomplish loyal customers and get new ones, mainly because it is not necessary acquire too much information about user in first meetings.

Our method is able to learn more about user when there is little information about him/her since it uses content information (and grades) in order to find the user's neighbors which is richer than using just the items' identifiers (and grades) as in CFA.

Another interesting result is that the observed standard deviation of the CFA and CMSA diminishes when the size of user profile is increased to 10. The reason for this behavior is that as more items are added in the user profile better will be the estimation of user's neighborhood and, consequently, better recommendations can be provided by the system to some users whose the profile was obscure when there was just 5 items. But, a more remarkable result is that **CSMA$_{k=50}$** reach low standard deviations implying more stable systems even in the presence of unusual users.

Notice that the CBCF system has the worst performance when there is 5 items in the user profile. This happens because the CBCF has too little information to estimate grades of all other items in database for some active user, which is a step of this algorithm. See that when there is more information about the user (10 items in user profile) the performance has been almost twice than in previous case.

As a final remark, it is not surprising that MSA algorithm would have worse prediction accuracies than CFA and CMSA because *CB* systems traditionally has worse performances than collaborative algorithms. In fact, the use of *CB* filtering is interesting to overcome some problems of *CF* systems, such as the *cold-start* one. In order to investigate this behavior another experiment would be necessary, but this is not our purpose in this work.

5 Final Comments and Conclusions

In the present work we describe a new *IF* method that uses the fundamental idea of Information Filtering Based on Modal Symbolic Objects approach [5] to build rich user profiles. We propose a suitable similarity function which are able to compare two *MS* user profiles. Throughout the proposed function we are able to select best neighbors of some active user in order to perform the *kNN-CF* algorithm.

The proposed method was evaluated in same conditions with other three information filtering algorithms. In order to perform this evaluation, it was prepared a experimental environment based on MovieLens and IMDB databases. The experimental environment was defined taking into account real scenarios of information systems, such as the lack of information about user in the beginning of system usage. An appropriate metric and methodology was used to measure the prediction accuracy of systems in *Find Good Items* task [7].

We show our new method improves the quality of recommendation lists when there is too little information about the user. If we suppose the user provide evaluations for 5 items in the first contact, which is very acceptable in real life, the system are able to supply good recommendation lists which can motivate the user to come back more times to the system rising fidelity.

Acknowledgments. The authors would like to thank CNPq (Brazilian Agency) for its financial support.

References

1. Balanovic, M. and Shoham, Y.: Fab: Content-based, collaborative recommendation. Communications of the ACM, Vol. 40 (1997) 88-89.
2. Bock, H.H. and Diday, E.: Analysis of Symbolic Data. Springer, Heidelberg (2000).
3. Breese, J., Heckerman, D., and Kadie, C: Empirical analysis of predictive algorithms for collaborative filtering. In Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (1998) 43-52.
4. Claypool, M., Gokhale, A., Miranda, T., Murnivok, P., Netes, D., and Sartin, M.: Combining Content-Based and Collaborative Filters in an Online Newspaper. ACM SIGIR Workshop on Recommender Systems, Berkeley, CA, August 19, 1999.
5. De Carvalho, F.A.T. and Bezerra, B.L.D.: Information Filtering based on Modal Symbolic Objects. Proceedings of the 26th Annual Conference of the Gesellschaft für Klassifikation (GfKl), Springer (2002) 395-404.
6. Herlocker, J., Konstan, J.A., Borchers, A., and Riedl, J.: An algorithmic framework for performing collaborative filtering. Proceedings of SIGIR (1999) 230-237.
7. Herlocker, J.L., Konstan, J.A., Terveen, L.G., and Riedl, J.: Evaluating Collaborative Filtering Recommender Systems. ACM Transactions on Information Systems, Vol. 22, Issue 1 (2004) 5-53.
8. Ichino, M., Yaguchi, H.: Generalized Minkowsky Metrics for Mixed Feature Type Data Analysis. IEEE Transactions on System, Man and Cybernetics, Vol. 24 (1994) 698–708.
9. Melville, P., Mooney, R.J., and Nagarajan, R.: Content-Boosted Collaborative Filtering for Improved Recommendations. Proceedings of the Eighteenth National Conference on Artificial Intelligence (2002) 187-192.

10. Popescul, A., Ungar, L.H., Pennock, D.M., and Lawrence, S.: Probabilistic Models for Unified Collaborative and Content-Based Recommendation in Sparse-Data Environments. 17th Conference on Uncertainty in Artificial Intelligence (2001).
11. Sarwar, B., Konstan, J., Borchers, A., Herlocker, J., Miller, B. and Riedl, J.: Using Filtering Agents to Improve Prediction Quality in the Grouplens Research Collaborative Filtering System. In Proceedings of the ACM Conference on Computer Supported Cooperative Work (1998) 345-354.
12. Schafer, J.B., Konstan, J.A., and Riedl, J.: E-Commerce Recommendation Applications. Data Mining and Knowledge Discovery, Vol. 5. (2001) 115-153.
13. Witten, I.H. and Frank, E.: Data Mining - Practical Machine Learning Tools and Techniques with Java Implementations. San Diego, CA: Morgan Kaufmann, (2000).

Machine Learning by Multi-feature Extraction Using Genetic Algorithms[1]

Leila S. Shafti and Eduardo Pérez

Universidad Autónoma de Madrid, E–28049 Madrid, Spain
leila.shafti@ii.uam.es, eduardo.perez@ii.uam.es
http://www.ii.uam.es/

Abstract. Constructive Induction methods aim to solve the problem of learning hard concepts despite complex interaction in data. We propose a new Constructive Induction method based on Genetic Algorithms with a non-algebraic representation of features. The advantage of our method to some other similar methods is that it constructs and evaluates a combination of features. Evaluating constructed features together, instead of considering them one by one, is essential when number of interacting attributes is high and there are more than one interaction in concept. Our experiments show the effectiveness of this method to learn such concepts.

1 Introduction

Learning algorithms based on similarity assume that cases belonging to the same class are located close to each other in the instance space defined by original attributes. So, these methods attain high accuracy on domains where the data representation is good enough to maintain the closeness of instances of the same class, such as those provided in Irvine databases [1]. But for hard concepts with complex interaction, each class is scattered through the space due to low-level representation [2, 3]. *Interaction* means the relation between one attribute and the target concept depends on another attribute. When the dependency is not constant for all values of the other attribute, the interaction is *complex* [4]. Figure 1 shows an example of interaction and complex interaction between two attributes in instance space. The complex interaction has been seen in real-world domains such as protein secondary structure [5].

Constructive induction (CI) methods have been introduced to ease the attribute interaction problem. Their goal is to automatically transform the original representation space of hard concepts into a new one where the regularity is more apparent [6, 7]. This goal is achieved by constructing *new* features from the given attribute set to abstract the interaction among several attributes into a new one.

Most CI methods apply a greedy search to find new features. But because of the attributes' interaction, the search space for constructing new features has

[1] This work has been partially supported by the Spanish Interdepartmental Commission for Science and Technology (CICYT), under Grant number TIC2002-1948.

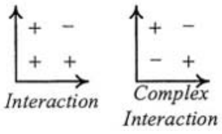


Fig. 1. Interaction and complex interaction in instance space

more variation. Recent works on problems with interaction [8, 3] show that a global search strategy such as Genetic Algorithms (GA) [9] is more likely to be successful in searching through the intractable and complicated search space [10].

Another limitation of many CI methods is the language they apply for representing constructed features. These methods use algebraic form of representing features. By *algebraic* form we mean features are shown by means of some algebraic operators such as arithmetic or Boolean operators. By contrast to this kind of representation, we have *non-algebraic* form of representation, which means no operator is used for representing the feature. For example, for a Boolean attribute set $\{X_1, X_2\}$ an algebraic feature like $(X_1 \wedge \overline{X}_2) \vee (\overline{X}_1 \wedge X_2)$ can be represented by a non-algebraic feature such as $\langle 0110 \rangle$, where the j^{th} element in $\langle 0110 \rangle$ represents the outcome of the function for j^{th} combination of attributes X_1 and X_2 acording to the truth table. This form of representation has been used in [11] and [12] for learning. We showed in [8] that a complex algebraic expression is required to capture and encapsulate the interaction into a feature, while non-algebraic representation reduces the difficulty of constructing complex features. For this reason, and in spite of their use of GA search strategy, some CI methods such as GCI [13], GPCI [14], Gabret [15] and the hybrid method of Ritthoff et al [16] fail when a high-order complex interaction exists among attributes.

There are very few methods that use non-algebraic form of representation; among them are MRP [17] and DCI [18]. These CI methods produced high accuracy on complex concepts with high interaction. However they have limitations and deficiencies. MRP represents features by sets of tuples obtained from training data using relational projection, and applies a greedy search in order to construct features. Each constructed feature is used as a condition to split data in two parts and then for each part, new features are generated. Therefore, only one feature is constructed and evaluated at a time. DCI applies GA to reduce the problem of local optima. However, it still evaluates features one at a time and is incapable of constructing more than one feature. Therefore, both methods construct and evaluate new features individually.

When the number of interacting attributes is high, the feature that encapsulates the interaction is complex and difficult to be constructed. A CI method must break down the complex interaction into several smaller features. Such set of features works as a theory of intermediate concepts that bridge the gap from the input data representation to the hard target concept. In that case each feature that partially shows the interaction, by itself, does not give enough information about concept and may be considered as an irrelevant feature. In order to see the goodness of new features, the CI method should evaluate the combination of features together as related parts of the theory. For this reason MRP, DCI, and other similar methods fail when number of interacting attributes grows and the interaction is complex (see Sect. 3).

In this paper, we will introduce MFE/GA, a Multi-feature Extraction method based on GA and non-algebraic form of representation, for constructing a set of new features to highlight the interaction that exists among attributes. Our experiments show the advantage of our method over other CI methods when number of interacting attributes grows, and the interaction is complex.

2 GA Design

MFE/GA uses GA to generate and combine features defined over different subsets of original attributes. The current version of the method only works with nominal attributes. Therefore, any continuous attribute should be converted to nominal attributes before running MFE/GA. The GA receives training data represented by original attributes set and finds subsets of interacting attributes and features representing the interaction. When the GA finishes, the new features are added to the attributes set and the new representation of data is given to a standard learner for learning. This section explains the system design.

2.1 Individuals Representation

For constructing new features we need to perform two tasks: finding the subset of interacting attributes and generating a function defined over each subset. Our individuals are sets of subsets of primitive attributes such as $Ind = \langle S_1, S_2, \ldots, S_k\rangle$ where $S_i \subset S$, $S_i \neq \emptyset$, and S is the set of original attributes.

Subsets in individuals are represented by bit-strings of length n, where n is the number of original attributes; each bit showing the presence or absence of the attribute in the subset. Therefore each individual is a bit-string of length $k.n$ $(k > 0)$ such as $Ind = \langle b_1, \ldots, b_n : b'_1, \ldots, b'_n : b''_1, \ldots, b''_n : \ldots\rangle$.

Since each individual has different number of subsets, the length of individuals is variable. To avoid unnecessary growth of individuals we have limited the number of subsets in individuals so that $k \leq 5$.

Each subset in individual is associated with a function that is extracted from data. Thus each individual is actually representing a set of functions such as $\{F_1, F_2, \ldots, F_k\}$, where F_i is a function defined over S_i. It is important to note that during mutation and crossover if a subset is changed, the associated function is also changed since a new F'_i is extracted for the new subset S'_i in the offspring.

The function F_i created for any given subset $S_i = \{X_{i1}, \ldots, X_{im}\}$ in an individual uses a non-algebraic form of representation (see Sect. 1). As explained in [8], this form of representation reduces the difficulty of constructing complex features. Each F_i is defined by assigning Boolean class labeles to all the tuples in the Cartesian product $X_{i1} \times \ldots \times X_{im}$. The class assigned to each tuple t depends on the training samples that *match* the tuple, that is, the training samples whose values for attributes in S_i are equal to the corresponding values in tuple t.

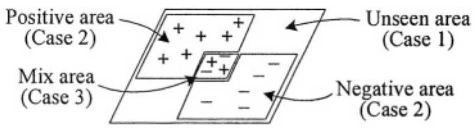


Fig. 2. Space of samples defined by attributes in subset S_i

More precisely, the class assigned depends on the class labels of all those training samples matching the tuple, as discussed next case by case:

Case 1. If there are no training samples matching t, a class label is assigned to $F_i(t)$ stochastically, according the class distribution in the training data.
Case 2. If all training samples matching t belong to the same class, this is the class assigned to $F_i(t)$.
Case 3. If there is a mixture of classes in the samples matching t, the class assigned to $F_i(t)$ depends on the numbers of tuples labelled by Case 2 as positive and negative, p_2 and n_2 respectively. In particular, if $p_2 = n_2 = 0$, the class is assigned stochastically (as in Case 1); if $p_2 > n_2$, the negative class is assigned; and otherwise, the positive class is assigned.

This procedure for extracting the definition of F_i from data partitions the subspace defined by S_i into four areas, as illustrated in Fig. 2. Each F_i identifies similar patterns (Case 2) of interaction among S_i and *compresses* them into the negative or postive area. The unseen area (Case 1) is covered by stochastically predicting the most frequent class. Note this covering of the unseen area means *generalization,* and thus may involve prediction errors.

As we will see next, the GA's fitness evaluation is applied to individuals composed of several F_i, each defined over a subset S_i. Thus, each individual encapsulates several interactions into features, and this allows the GA to simultaneously construct and evaluate features, which turns out to be essential when several high-order interactions exist in data.

2.2 Fitness Function

After feature extraction, for each $Ind = \langle S_1, \ldots, S_k \rangle$, data are projected onto the set of new features $\{F_1, \ldots, F_k\}$ and the goodness of the individual is evaluated by the following formula:

$$Fitness(Ind) = \frac{min(|\pi^+ - \pi^-|, |\pi^- - \pi^+|)k + \|\pi^+ \cap \pi^-\|(k+1)}{r(k+1)} + \frac{\sum |S_i|}{k|S|} \qquad (1)$$

where π^+ is set of positive tuples and π^- is set of negative tuples obtained by projecting data into $\{F_1, \ldots, F_k\}$, r is the total number of tuples in training data, the single bars $|z|$ denote the number of attributes (or tuples) in subset (or relation) z, and the double bars $\|p\|$ denotes the number of examples in training data that match with the tuples in relation p. The objective of GA is to minimize the value of *Fitness(Ind)*. The first term in this formula estimates how good is the set of newfeatures for classifying data. It is divided by $k+1$ to favor individuals with larger number of subsets. The aim is to prefer several simple features to few complex features. The complexity of features is evaluated in the last term by measuring the fraction of attributes participating in constructing features.

To reduce overfitting, we use 90% of training data for generating functions and all training data for fitness evaluation. Aside from that, the empirical evaluation of the system will be based on unseen data (see Sect. 3).

2.3 GA Operators

The genetic operators have the role of converging GA to optimal solution. We use mutation and crossover operators. Our objective is to generate different subsets of attributes with their associated functions and combine them to eventually find the group of functions defined over subsets of interacting attributes. To achieve this objective, we apply operators in two levels: attributes level to generate different subsets and features; and, subsets level to make different combination of subsets and features. Therefore we have two types of mutation and two types of crossover, illustrated in Fig. 3 where colons are used to seperate subsets of attributes, bars mark the crossover points, and underlined are genes in the parents that will be substituted by the operator.

Mutation Type-1–Mutation in Attributes Level: This operator is performed by considering the individual as a bit-string of size $k.n$ where k is the number of subsets and n is the number of original attributes. The traditional mutation is applied over the bit-string to flip bits of the string. By flipping a bit, we eliminate/add an attribute from/to any subset in any given individual.

Mutation Type-2–Mutation in Subsets Level: For this operator an individual is considered as a sequence of subsets and mutation is performed over any subset as whole, by replacing the subset with a new generated subset. Therefore, after this operation, some subsets are eliminated from the given individual and new subsets are added to produce a new combination of subsets.

Crossover Type-1–Crossover in Attributes Level: This operator applies classical two-point crossover considering the individual as a bit-string. Its aim is to generate new subsets by recombining segments of subsets from the parents. The two crossing points in the first parent are selected randomly. On the second parent, the crossing points are selected randomly, subject to the restriction that they must have the same distance from the subsets boundary in bit-string

Mutation in Attributes Level	**Mutation in Subsets Level**
Parent1 = $\langle 1\underline{0}01001\underline{0}{:}01\underline{0}10100{:}000\underline{1}0111 \rangle$	Parent2 = $\langle S_1, \underline{S_2}, S_3, S_4, S_5 \rangle$
Child1 = $\langle 1\overline{1}01001\overline{1}{:}01\overline{1}10100{:}000\overline{0}0111 \rangle$	Child2 = $\langle S_1, \overline{S_2'}, S_3, S_4, S_5 \rangle$
Crossover in Attributes Level	**Crossover in Subsets Level**
Parent3 = $\langle 1001 \vert \underline{0010{:}010} \vert 10100{:}00010111 \rangle$	Parent5 = $\langle S_{11}, \underline{S_{12}} \rangle$ Mask1= $\langle 10 \rangle$
Parent4 = $\langle 0010 \vert \underline{1011{:}10010001{:}111} \vert 01100 \rangle$	Parent6 = $\langle S_{21}, \underline{S_{22}, S_{23}}, S_{24}, \underline{S_{25}} \rangle$ Mask2= $\langle 01101 \rangle$
Child3 = $\langle 10011011{:}10010001{:}11110100{:}00010111 \rangle$	Child5 = $\langle S_{11}, S_{22}, S_{23}, S_{25} \rangle$
Child4 = $\langle 00100010{:}01001100 \rangle$	Child6 = $\langle S_{12}, S_{21}, S_{24} \rangle$

Fig. 3. GA operators

representation as they had in the first parent [19]. Depending on where the crossing points are situated this operator may generate new subsets from subsets of parents and/or recombine subsets. This operator may change the length of the individual but we impose the limitation of $k \leq 5$; otherwise, new crossing points are selected until the produced offspring have $k \leq 5$ subsets.

Crossover Type-2–Crossover in Subsets Level: The aim of this operator is to generate different combinations of subsets by exchanging subsets of parents. It considers individuals as sequence of subsets and performs uniform crossover. Two crossover masks are generated randomly to define the cutting points. This operator may change the length of the individuals and therefore has the restriction of $k \leq 5$ same as above. Crossover Type-2 only recombines subsets and does not generate any new subset.

3 Empirical Analyses

We analyzed MFE/GA by empirically comparing it with other similar methods. For implementing GA we used PGAPack Library [20] with default parameters except those indicated in Table 1. The type of mutation and crossover operators (see Sect. 2.3) is specified by flipping a coin when individuals are selected for reproduction. We used 90% of training data for constructing functions and all training data for evaluating the constructed feature using (1).

To compare MFE/GA with other CI methods, we performed experiments over synthetic problems. Since our objective is to learn concepts with more than one interaction, artificial problems of this kind were selected for experiments. These problems were used as prototypes to exemplify complex interactions in real-world hard problems. Therefore, similar results are expected for real-world problems, where the main difficulty is complex interaction.

These concepts are defined over 12 Boolean attributes $a_1, \ldots, a_{12}$ as follows:

- $cp(i,j) = Parity(a_i, \ldots, a_6) \wedge Parity(a_7, \ldots, a_j)$
- $cdp(i,j) = Parity(a_i, \ldots, a_4) \wedge (Parity(a_{(i+j)/2}, \ldots, a_8) \vee Parity(a_j, \ldots, a_{12}))$
- $P(3,6) \wedge (l) = Parity(a_3, \ldots, a_6) \wedge$ (exactly l attributes in $\{a_7, \ldots, a_{12}\}$ are true)
- $P(3,6) \vee (l) = Parity(a_3, \ldots, a_6) \vee$ (exactly l attributes in $\{a_7, \ldots, a_{12}\}$ are true)

All above concepts have complex interaction among attributes that can be represented by several smaller interactions.

For each concept MFE/GA was run 20 times independently using 5% of shuffled data for training and the rest for final evaluation. When MFE/GA is finished, its performance was evaluated by the accuracy of C4.5 [21] on modified data after adding constructed features, using 95% unseen data as test data.

Table 1. GA's modified parameters

GA Parameter	New Value	GA Parameter	New Value
Population Size	100	Mutation Probability	0.01
Max Iteration	350	Num. of Strings to be Replaced	90
Max No Change Iteration	100		

Table 2. Summary of CI methods

CI Method	Algebraic Representation	Feature Evaluation	Genetic Search	CI Method	Algebraic Representation	Feature Evaluation	Genetic Search
Fringe	Yes	No Individually	No	MRP	No	Individually	No
Grove	Yes	Individually	No	DCI	No	Individually	Yes
Greedy3	Yes	Individually	No	MFE	No	No Individually	Yes
LFC	Yes	Individually	No	GA			

Table 3. System comparison by average accuracy

Concept	Num. of Relevant Attributes	Prior Best Result	MRP	DCI + C4.5	MFE/GA + C4.5
cp(4,9)	6	86.1 (Fringe)	99.0 (1.6)	97.2 (1.8)	98.9(4.0)
cp(3,10)	8	73.4 (C4.5-rules)	89.9 (1.2)	81.6 (1.6)	**95.7 (8.9)**
cp(2,11)	10	73.9 (C4.5)	91.7 (5.7)	68.0 (1.2)	**97.1 (4.8)**
cdp(3,11)	6	98.4 (Fringe)	97.3 (3.9)	97.6 (1.6)	99.2 (3.1)
cdp(2,10)	9	78.1 (Fringe)	92.0 (6.5)	67.7 (2.0)	86.6 (10.3)
cdp(1,9)	12	62.5 (C4.5-rules)	**81.3 (3.0)**	56.4 (2.5)	71.1 (7.4)
$P(3,6)\wedge(2)$	10	88.1 (C4.5)	90.4 (4.5)	81.0 (1.5)	**95.9 (2.9)**
$P(3,6)\wedge(3)$	10	83.4 (C4.5)	87.6 (5.4)	75.9 (2.4)	**94.1 (5.4)**
$P(3,6)\wedge(3$ or $2)$	10	70.4 (Fringe)	79.0 (2.3)	65.4 (2.6)	**90.8 (5.6)**
$P(3,6)\vee(2)$	10	70.5 (Fringe)	**97.1 (2.9)**	57.9 (2.2)	90.3 (6.6)
$P(3,6)\vee(3)$	10	68.0 (Fringe)	95.9 (4.4)	59.3 (1.7)	92.1 (7.1)
$P(3,6)\vee(3$ or $2)$	10	75.1 (Fringe)	80.4 (6.0)	69.5 (2.2)	**92.5 (7.5)**

We compared MFE/GA with C4.5 and C4.5-Rules [21], which are similarity-based learners, Fringe, Grove and Greedy3 [22], and LFC [23], which are greedy-based CI methods that use algebraic representation of features and MRP [17] which is a greedy CI method with a non-algebraic form of representation (see Sect. 1). Among CI methods only Fringe constructs several features at once. Other methods consider features one at a time. We also compared MFE/GA with DCI [18], which applies GA to reduce the problem of local minima (see Sect. 1). This method uses a feature representation similar to MFE/GA. However, it constructs and evaluates one feature during each generation. Therefore, when an interaction among a large set of attributes exists in the concept, this method fails to learn the concept. Table 2 summarizes the CI methods that are used in our experiments.

Table 3 gives a summary of MFE/GA's average accuracy over 20 runs and its comparison with other systems' average accuracy. In the third column of the table, we show the best results among C4.5, C4.5-Rules, Fringe, Grove, Greedy3 and LFC, as reported in [17]. Numbers between parentheses indicates standard deviation. Bolds means with a significant level of 0.05, this accuracy is the best between MRP and MFE/GA.

In these experiments, LFC, Greedy3 and Grove never appear as the best competitor among CI methods with algebraic representation. This may be due to the fact that they evaluate each proposed feature individually. When more than one interaction exist among attributes, a feature that encapsulates a single interaction may not provide, by itself, enough information about the final target concept, and therefore, is evaluated as an irrelevant feature.

MRP gives better result than other greedy methods because of its non-algebraic form of representing features. When high interaction exists among attributes, a more complex algebraic feature is needed to abstract the interaction. However, a non-algebraic representation may capture the structure of the interaction and abstract it more easily.

Experiments in [18] show that the genetic-based method, DCI, outperforms MRP when the concept consists of only one complex interaction over large number of attributes, e.g. $Parity(a_1, \ldots, a_8)$. But for concepts with more than one interaction, like those in Table 3, DCI cannot achieve good accuracy because it is incapable of constructing more than one feature. It finds a subset of interacting attributes and constructs one feature to encapsulate the interaction. But as the interaction is complex, the constructed feature is not good enough to outline all the interaction and, hence, MRP's greediness outperforms this method.

However, while the number of interacting attributes grows, the concept becomes more complex to be learned. Since MRP constructs and evaluates features separately with a greedy search, its performance decays. MFE/GA successfully breaks down the interaction over relevant attributes into two or more interactions over smaller subsets of attributes using a global search; and therefore, it gives better accuracy than other methods in most concepts of Table 3.

The synthetic concepts $cdp(i, j)$ illustrates well the different behaviors and advantages of MRP and MFE/GA. Each of the concepts *cdp*(1, 9), *cdp*(2,10) and *cdp*(3, 11) involves three parity relations combined by simple interactions (Conjunction and Disjunction of Parity). The three concepts differ in the degree of parity involved (4, 3, and 2, respectively) but perhaps more importantly, they also differ in the ratio of relevant attributes (12/12, 9/12, and 6/12, respectively). This is the reason why, obviously, all results in Table 3 indicate that *cdp*(1, 9) is the most difficult concept to learn: it has no irrelevant attributes that when projected away allow the complex substructures of the concept to became apparent, when learning from only 5% of data.

This affects MFE/GA in a higher degree than it affects MRP, probably due to differences between both systems' biases. In particular, considering *cdp*(1,9), MRP's focus on learning one single best relation probably guides learning toward $Parity(a_1, \ldots, a_4)$. As stated above, the interactions that combine parity features in this concept are simple (conjunction and disjunction). So MRP easily finds its way toward that parity feature. Had it used only that single feature to classify unseen data, it would have obtained even higher accuracy than it does (up to 87%). Perhaps, due to overfitting, MRP's heuristic function does not allow the system to reach such theoretically best possible performance on this concept. However, MRP's bias gets it closer to the goal than MFE/GA.

MFE/GA's bias is, in some sense, opposite to MRP's. It focuses on learning multiple features at once to evaluate them in combination. This higher flexibility in searching a large and complex feature space makes the system more dependent on data quality (since features are extracted from training data). Thus, MFE/GA exploits better than MRP the redundancy in data for *cdp*(3, 11). Since concepts *cdp*(2, 10) and *cdp*(3, 11) are defined, respectively, over 9 and 6 attributes of a total of 12 attributes, the 5% training data is likely to contain repeated parts of the concept structure, that become more apparent when projecting away the irrelevant attributes. However, this does not make these concepts easy to learn by non-CI methods, due to the complexity of the interactions involved. MRP tries to learn these two concepts as a single relation each, whereas MFE/GA beats it by seeing the multiple features involved at once. On the other hand, for *cdp*(1, 9), all 12 attributes are relevant, and so there is little or no redundancy in the 5% training data. Therefore, MFE/GA's more flexible search gets trapped in a local minima, overfitting data, whereas MRP is favored by its strong bias for *one best* relation, which in this case does indeed exists and it is easy to find.

It is important to note that in all experiments MFE/GA generates close approximations to the sub-interactions; excluding experiments over *cdp*(1, 9), in more than 86% of experiments our method successfully finds the exact subsets of interacting attributes.

Our use of synthetic concepts allows us to analyze system behavior deeply before moving on to try to solve real-world problems with difficulties similar to those exemplified by these synthetic concepts.

4 Conclusion

This paper has presented MFE/GA, a new CI method for constructing a combination of features to highlight the relation among attributes when high complex interaction exists in the target concept. The method applies a non-algebraic form of representing features, which is more adequate for constructing features when interaction is complex.

The genetic approach provides the ability of constructing and evaluating several features at once. Most CI methods consider features individually. When the complex interaction is of higher order and need to be represented by more than one feature, each feature by itself may not give enough information about the interaction and therefore is evaluated as irrelevant. The new features should be considered in combination when evaluated, as if they build up a theory composed of intermediate concepts that bridge the gap between a primitive low level representation and a high level complex concept.

Our experiment shows the advantage of non-algebraic representation of features over algebraic representation. Also it shows that CI methods that consider features individually fail to learn concepts when number of interacting attributes grows. Our method with GA-based global search and non-algebraic representation successfully finds the combination of features that represent interaction and outperforms other CI methods when the interaction becomes more complex.

References

1. Blake, C.L., Merz, C.J.: UCI Repository of Machine Learning Databases, [http://www.ics.uci.edu/~mlearn/MLRepository.html]. University of California, Department of Information and Computer Science, Irvine,CA (1998)
2. Rendell, L.A., Seshu, R.: Learning hard concepts through constructive induction: framework and rationale. Computational Intelligence, **6**:247–270 (Nov., 1990)

3. Freitas, A.: Understanding the crucial role of attribute interaction in data mining. Artificial Intelligence Review, **16(3)**:177–199 (Nov., 2001)
4. Pérez, E.: Learning despite complex interaction: an approach based on relational operators, PhD Thesis, University of Illinois, Urbana-Champaign (1997)
5. Qian, N., Sejnowski, T.J.: Predicting the secondary structure of globular proteins using neural network models. Molecular Biology, **202**:865–884 (Aug., 1988)
6. Dietterich, T.G., Michalski, R.S.: Inductive learning of structural description: evaluation criteria and comparative review of selected methods. Artificial Intelligence, **16(3)**:257–294 (Jul., 1981)
7. Aha, D.W.: Incremental constructive induction: an instance-based approach. In Proc. of the Eighth International Workshop on Machine Learning, pages 117–121, Evanston, Illinois (1991) Morgan Kaufmann
8. Shafti, L.S., Pérez, E.: Genetic approach to constructive induction based on non-algebraic feature representation, In Proc. of the Fifth International Symposium on Intelligent Data Analysis, pages 509–520, Berlin, Heidelberg (2003) LNCS
9. Holland, J.H.: Adaptation in Natural and Artificial Systems. The University of Michigan Press, Ann Arbor, Michigan (1975)
10. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag, Berlin, Heidelberg, New York (1999)
11. Samuel, A.: Some studies in machine learning using the game of checkers II: recent progress, IBM J. Res. Develope. **11**:601–617 (1967)
12. Zupan, B., Bratko, I., Bohanec, M., Demsar, J.: Function decomposition in machine learning, LNAI **2049**:71-101 (2001)
13. Bensusan, H. Kuscu I.: Constructive induction using genetic programming. In Proc. of ICML'96 Workshop of Evolutionary Computing and Machine Learning, Bari, Italy (1996) T. Fogarty and G. Venturini (Eds.)
14. Hu, Y.: A genetic programming approach to constructive induction. In Proc. of the Third Annual Genetic Programming Conference, pages 146-157, Madison, Wisconsin (1998) Morgan Kauffman
15. Vafaie, H., De Jong, K.: Feature space transformation using genetic algorithms. IEEE Intelligent Systems and Their Applications, **13(2)**:57–65 (Mar.–Apr., 1998)
16. Ritthoff, O., Klinkenberg, R., Fischer, S., Mierswa, I.: A hybrid approach to feature selection and generation using an evolutionary algorithm, In Proc. of UK Workshop on Computational Intelligence (Sep., 2002) The University of Birmingham
17. Pérez, E. Rendell, L.A.: Using multidimensional projection to find relations. In Proc. of the Twelfth International Conference on Machine Learning, pages 447–455, Tahoe City, California (Jul. 1995) Morgan Kaufmann
18. Shafti, L.S., Pérez, E.: Constructive induction using non-algebraic feature representation, In Proc. of the Third IASTED International Conference on Artificial Intelligence, pages 134–139, Benalmadena, Spain (2003) Acta Press
19. DeJong, K. A., Spears, W.M., Gordon, D.F.: Using Genetic Algorithms for Concept Learning. Machine Learning, **13**:161–139 (1993)
20. Levine, D.: Users guide to the PGAPack parallel genetic algorithm library. Technical Report ANL-95/18, Argonne National Laboratory (Jan. 1996)
21. Quinlan, R.J.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo, California (1993)
22. Pagallo G., Haussler, D.: Boolean feature discovery in empirical learning. Machine Learning, **5**:71–99 (1990)
23. Ragavan, H., Rendell, L.A.: Lookahead feature construction for learning hard concepts. In Proc. of the Tenth International Conference on Machine Learning, pages 252–259, Amherst, Massachusetts (1993) Morgan Kaufmann

Assignment of Semantic Roles Based on Word Sense Disambiguation

Paloma Moreda Pozo, Manuel Palomar Sanz, and Armando Suárez Cueto

Grupo de investigación del Procesamiento del Lenguaje y Sistemas de Información, Departamento de Lenguajes y Sistemas Informáticos, Universidad de Alicante, Alicante, Spain
{moreda, mpalomar, armando}@dlsi.ua.es

Abstract. In order to achieve high precision Question Answering Systems or Information Retrieval Systems, the incorporation of Natural Language Processing techniques are needed. For this reason, in this paper a method to determine the semantic role for a constituent is presented. The goal of this is to integrate the method in a Question Answering System and in an Information Retrieval System. So, several experiments about the Semantic Role method, named SemRol, are shown.

1 Introduction

One of the challenges of applications such as Information Retrieval (IR) or Question Answering (QA), is to develop high quality systems (high precision IR/QA). In order to do this, it is necessary to involve Natural Language Processing (NLP) techniques in this kind of systems. Among the different NLP techniques which would improve Information Retrieval or Question Answering systems it is found Word Sense Disambiguation (WSD) and Semantic Role Labeling (SRL). In this paper a method of Semantic Role Labeling using Word Sense Disambiguation is presented. This research is integrated in the project R2D2[1].

A semantic role is the relationship between a syntactic constituent and a predicate. For instance, in the next sentence

(E0) The executives gave the chefs a standing ovation

The executives has the Agent role, *the chefs* the Recipient role and *a standing ovation* the Theme role.

The problem of the Semantic Role Labeling is not trivial. In order to identify the semantic role of the arguments of a verb, two phases have to be solved, previously. Firstly, the sense of the verb is disambiguated. Secondly, the argument boundaries of the disambiguated verb are identified.

[1] This paper has been supported by the Spanish Government under project "R2D2: Recuperación de Respuestas de Documentos Digitalizados" (TIC2003-07158-C04-01).

First of all, the sense of the verb has to be obtained. Why is necessary to disambiguate the verb? Following, an example shows the reason for doing so.

(E1) John gives out lots of candy on Halloween to the kids on his block

(E2) The radiator gives off a lot of heat

Depending on the sense of the verb a different set of roles must be considered. For instance, Figure 1 shows three senses of verb *give* (give.01, give.04, and give.06)) and the set of roles of each sense. So, sentence (E0) matches with sense give.01. Therefore, roles *giver, thing given* and *entity given to* are considered. Nevertheless, sentence (E1) matches with sense give.06 and sentence (E2) matches with sense give.04. Then, the sets of roles are *(distributor, thing distributed, distributed)* and *(emitter, thing emitted),* respectively. In sentence (El), *John* has the distributor role, *lots of candy* the thing distributed role, *the kids on his block* the distributed role and *on Halloween* the temporal role. In sentence (E2), *the radiator* has the emitter role and *a lot of heat* the thing emitted role. These examples show the relevance of WSD in the process of assignment of semantic roles.

```
<roleset id="give.01" name="transfer"> <roles>
  <role n="0"  descr="giver"  vntheta="Agent"/>
  <role n="1"  descr="thing given"  vntheta="Theme"/>
  <role n="2"  descr="entity given  vntheta="Recipient""/>
</roles>

<roleset id="give.04" name="emit"> <roles>
  <role n="0"  descr="emitter"/>
  <role n="1"  descr="thing emitted"/>
</roles>

<roleset id="give.06" name="transfer"> <roles>
  <role n="0"  descr="distributor"/>
  <role n="1"  descr="thing distributed"/>
  <role n="2"  descr="distributed"/>
</roles>
```

Fig. 1. Some senses and roles of the frame *give* in PropBank [13]

In the second phase, the argument boundaries are determined. For instance, in the sentence (E0), the argument boundaries recognized are

[The executives] gave [the chefs] [a standing ovation]

Once these two phases are applied, the assignment of semantic roles can be carried out.

To achieve high precision IR/QA systems, recognizing and labeling semantic arguments is a key task for answering “Who”, “When”, “What”, “Where”, “Why”, etc. For instance, the following questions could be answered with the sentence (E0). The Agent role answers the question (E3) and the Theme role answers the question (E4).

(E3) Who gave the chefs a standing ovation?

(E4) What did the executives give the chefs?

These examples show the importance of semantic roles in applications such as Information Retrieval or Question Answering.

Currently, several works have tried using WSD or Semantic Role Labeling in IR or QA systems, unsuccessfully. Mainly, it is due to two reasons:

1. The lower precision achieved in these tasks.
2. The lower portability of these methods.

It is easy to find methods of WSD and Semantic Role Labeling that work with high precision for a specific task or specific domain. Nevertheless, this precision drops when the domain or the task are changed. For these reasons, this paper is about the problem of a Semantic Role Labeling integrated with WSD system. A method based on a corpus approach is presented and several experiments about both, WSD and Semantic Role Labeling modules, are shown. Shortly, a QA system with this Semantic Role Labeling module using WSD will be developed in the R2D2 framework.

The remaining paper is organized as follows: section 2 gives an idea about the state-of-art in automatic Semantic Role Labeling systems in the subsection 2.1. Afterwards, the maximum entropy-based method is presented in subsection 2.2. Then, some comments about experimental data, and an evaluation of our results using the method, are presented in sections 3 and 4, respectively. Finally, section 5 concludes.

2 The SemRol Method

The method, named SemRol, presented in this section consists of three phases:

1. Verb Sense Disambiguation phase (VSD)
2. Argument Boundaries Disambiguation phase (ABD)
3. Semantic Role Disambiguation phase (SRD)

These phases are related since the output of VSD phase is the input of ABD phase, and the output of ABD phase is the input of SRD phase. So, the success of the method depends on the success of the three phases.

Both, Verb Sense Disambiguation phase and Semantic Role Disambiguation phase are based on Maximum Entropy (ME) Models. Argument Boundaries Disambiguation and Semantic Role Disambiguation phases take care of recognition and labeling of arguments, respectively. VSD module means a new phase in the task. It disambiguates the sense of the target verbs. So, the task turns more straightforward because semantic roles are assigned to sense level.

In order to build this three-phase learning system, training and development data set are used. It is used PropBank corpus [13], which is the Penn Treebank corpus [11] enriched with predicate-argument structures. It addresses predicates expressed by verbs and labels core arguments with consecutive numbers (A0 to A5), trying to maintain coherence along different predicates. A number of adjuncts, derived from the Treebank functional tags, are also included in PropBank annotations.

A previous approximation of this method is presented in [12]. In this paper, the main features of VSD and SRD phases and the experiments that prove the results are shown.

2.1 Background

Several approaches [3] have been proposed to identify semantic roles or to build semantic classifier. The task has been usually approached as a two phase procedure consisting of recognition and labeling arguments.

Regarding the learning component of the systems, several systems can be found in CoNLL 2004 shared task [1]. For instance, Maximum Entropy models ([2]; [10]), Brill's Transformation-based Error-driven Learning ([8]; [18]), Memory-based Learning ([17]; [9]), vector-based linear classifiers ([7]; [14]), Voted Perceptrons [4] or, SNoW, a Winnow-based network of linear separators [15].

In these systems only partial syntactic information, i.e., words, part-of-speech (PoS) tags, base chunks, clauses and named entities, is used.

2.2 The Core of SemRol Method

The method consists of three main modules: i) Verb Sense Disambiguation (VSD) Module, ii) Argument Boundaries Disambiguation (ABD) Module, and iii) Semantic Role Disambiguation (SRD) Module.

First of all, the process to obtain the semantic role needs the sense of the target verb. After that, several heuristics are applied in order to obtain the arguments of the sentence. And finally, the semantic roles that fill these arguments are obtained.

Verb Sense Disambiguation Module. This module is based on the WSD system developed by [16]. It is based on conditional ME probability models.

The learning module produces classifiers for each target verb. This module has two subsystems. The first subsystem processes the learning corpus in order to define the functions that will apprise the linguistic features of each context. The second subsystem of the learning module performs the estimation of the coefficients and stores the classification functions.

The classification module carries out the disambiguation of new contexts using the previously stored classification functions. When ME does not have enough information about a specific context, several senses may achieve the same maximum probability and thus the classification cannot be done properly. In these cases, the most frequent sense in the corpus is assigned. However, this heuristic is only necessary for a minimum number of contexts or when the set of linguistic attributes processed is very small.

The set of features defined for the training of the system is based on words, PoS tags, chunks and clauses in the local context.

Argument Boundaries Disambiguation Module. After determining the sense for every target verb of the corpus, it is necessary to determine the argument boundaries of those verbs. In order to do so, left and right arguments have to be considered. Left argument is the noun phrase at the left of the target verb, and right argument is the noun phrase at the right of the target verb. Besides, if exists a prepositional phrase close together the right noun phrase, it will be considered a second right argument. In any case, the phrases must belong to the same clause as the target verb. So, in the sentence the target verb is *narrow,* the left argument is *the current account deficit* and right arguments are *only 1.8 billion* and *in September.*

(E5) The current account deficit will narrow to only 1.8 billion in September

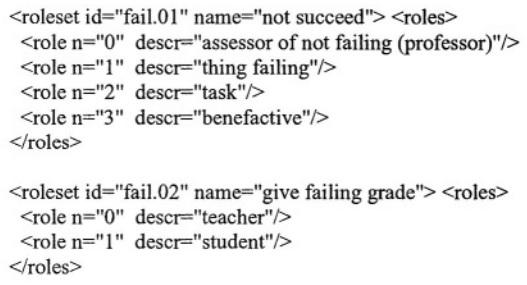
Finally, the verbal phrase of the target verb is considered as the verb argument, and modal verbs and particles *not* and *n't* in the verbal phrase of the target verb, as arguments. For instance, in the previous sentence, *will* is considered an argument.

It is expected that the number of successes in left arguments, modal arguments and negative arguments, will be high and it will not account for much error. However, the results in right arguments will be probably lower. In future works we will take interest in determining the arguments of the verbs using a machine learning strategy, such as a maximum entropy conditional probability method, or a support vector machines method [6]. This strategy will allow us to determine the argument boundaries more accurately.

Semantic Role Disambiguation Module. Finally, the role for each target verb depending on sense will be determined. This task uses a conditional ME probability model. This one is like the method used in WSD task. In this case, features are extracted for each argument for every target verb. These features are used to classify those arguments. Instead of working with all roles [5], in this classification, the classes considered will be the roles of each sense of each verb. It increases the total number of the classes for the full task on SRD, but it reduces the partial number of classes that are taken into account in each argument, considerably. In the sentence (E6), the sense of fail is 01, so, the classes of the roles 0,1,2,3, of fail.01 have just been considered, however the roles 0,1 of fail.02 have not been considered. It is possible to do this because the sense of every target verb was determined in the VSD module. Figure 2 shows the roles of *fail* verb.

(E6) Confidence in the pound is widely expected to take another sharp dive if trade figures for September, due for release tomorrow, fail to show a substantial improvement from July and August's near-record deficits

The set of features defined for the training of the system is based on words, PoS tags, chunks and clauses in the local context.



```
<roleset id="fail.01" name="not succeed"> <roles>
  <role n="0"  descr="assessor of not failing (professor)"/>
  <role n="1"  descr="thing failing"/>
  <role n="2"  descr="task"/>
  <role n="3"  descr="benefactive"/>
</roles>

<roleset id="fail.02" name="give failing grade"> <roles>
  <role n="0"  descr="teacher"/>
  <role n="1"  descr="student"/>
</roles>
```

Fig. 2. Senses and roles of the frame *fail* in PropBank

3 Experimental Data

Our method has been trained and evaluated using the PropBank corpus [13], which is the Penn Treebank [11] corpus enriched with predicate-arguments structures. To be precise, the data consists of sections of the Wall Street Journal. Training set matches with sections 15-18 and development set matches with section 20.

PropBank annotates the Penn Treebank with arguments structures related to verbs. The semantic roles considered in PropBank are the following [3]:

- Numbered arguments (A0-A5, AA): Arguments defining verb-specific roles. Their semantic depends on the verb and the verb usage in a sentence, or verb sense. In general, A0 stands for the *agent* and A1 corresponds to the *patient* or *theme* of the proposition, and these two are the most frequent roles. However, no consistent generalization can be made across different verbs or different senses of the same verb. PropBank takes the definition of verb senses from VerbNet, and for each verb and each sense defines the set of possible roles for that verb usage, called roleset.
- Adjuncts (AM-): General arguments that any verb may take optionally. There are 13 types of adjuncts:
  - AM-LOC: location
  - AM-EXT: extent
  - AM-DIS: discourse marker
  - AM-ADV: general-porpouse
  - AM-NEC: negation marker
  - AM-MOD: modal verb
  - AM-CAU: cause
  - AM-TEMP: temporal
  - AM-PRP: purpose
  - AM-MNR: manner
  - AM-DIR: direction
- References (R-): Arguments representing arguments realized in other parts of the sentence. The role of a reference is the same than the role of the

Table 1. Results on the development set

VSD	Successes	3650	Precision	**0.88**
	Fails	486	Recall	0.85
	No disambiguated	169	F1	0.86
ABD	Successes	4426	Precision	**0.51**
	Fails	4201	Recall	0.40
	No disambiguated	2494	F1	0.45
SRD	Successes	5085	Precision	**0.48**
	Fails	5464	Recall	0.46
	No disambiguated	572	F1	0.47

referenced argument. The label is an R-tag preceded to the label of the referent, e.g. R-A1.

– Verbs (V): Participant realizing the verb of the proposition.

Training data consists of 8936 sentences, with 50182 arguments and 1838 distinct target verbs. Development data consists of 2012 sentences, with 11121 arguments and 978 distinct target verbs.

Apart from the correct output, both datasets contain the input part of the data: PoS tags, chunks and clauses. Besides, the sense of verb is available if the word is a target verb.

4 Results and Discussion

Following, the results of the three modules are shown in Table 1. These results have been obtained on the development set. Modules have been evaluated based on precision, recall and F1 measure. In each case, precision is the proportion of senses, arguments or roles predicted by the system which are correct; and recall is the proportion of correct senses, correct arguments or correct roles which are predicted by each module. That is, the senses, arguments and roles not desambiguated are not considered when computing precision but they are considered when computing recall. F1 computes the harmonic mean of precision and recall: $F_{\beta=1}=(2pr)/(p+r)$.

4.1 VSD Module Results

In this experiment one set of features have just been considered, features about *content word in a sentence.* Table 1 shows that 3650 verbs have been disambiguated successfully, and 655 unsuccessfully. From these, 169 are due to no disambiguated verbs and 486 to mistakes in the disambiguation process. As a result, a precision of 88% is obtained. These results show the goodness of the ME module and reveal that the ME module is correctly defined. Besides, it is expected that the tuning with the others set of features (see section 2.2) will improve the results.

Table 2. Results on the development set. SRD module

	Precision	Recall	$F_{\beta=1}$		Precision	Recall	$F_{\beta=1}$
A0	47.22%	44.56%	45.85	AM-MNR	66.29%	17.66%	27.90
A1	44.99%	69.12%	54.50	AM-MOD	82.35%	39.59%	53.47
A2	52.26%	26.62%	35.28	AM-NEG	80.00%	64.12%	71.19
A3	30.16%	12.75%	17.92	AM-PNC	39.47%	15.00%	21.74
A4	64.52%	13.61%	22.47	AM-PRD	0.00%	0.00%	0.00
A5	100.00%	25.00%	40.00	AM-PRP	0.00%	0.00%	0.00
AM-ADV	21.93%	7.10%	10.73	AM-REC	0.00%	0.00%	0.00
AM-CAU	0.00%	0.00%	0.00	AM-TMP	61.98%	21.48%	31.90
AM-DIR	55.56%	8.33%	14.49	R-A0	0.00%	0.00%	0.00
AM-DIS	86.44%	25.00%	38.78	R-A1	0.00%	0.00%	0.00
AM-EXT	70.00%	14.29%	23.73	R-A2	0.00%	0.00%	0.00
AM-LOC	44.44%	22.61%	29.97	R-AM-LOC	100.00%	25.00%	40.00
R-AM-TMP	33.33%	33.33%	33.33	V	97.44%	97.44%	97.44
				all	**61.92%**	**59.62%**	**60.75**
				all−{V}	**47.42%**	**44.98%**	**46.17**

4.2 ABD Module Results

In this case, a total of 4426 arguments have been detected successfully, but 6695 have been erroneously detected or missing. Therefore, the precision of ABD module is 51%.

In any case, the experiments have been done assuming correct senses for target verbs. By means of this, the independence of ABD module in relation to VSD module has been evaluated.

These results confirm the need for determining the arguments of the verbs by defining new heuristics or using a machine learning strategy.

4.3 SRD Module Results

In order to evaluate this module, correct senses of the verbs and correct argument boundaries have been presumed. So, SRD module has been tested independently of VSD and ABD modules.

Table 1 shows a precision of 48%. For further details, the precision for each kind of argument is shown in Table 2. Besides, if verb argument is considered, precision goes up to 62%. These results show that the ME module is correctly defined. However, it is need a tuning phase in order to improve them. Besides, a precision of 0,00% in several *R-* arguments shows the need for a co-reference resolution module.

5 Conclusions and Working in Progress

In this paper, a Semantic Role Labeling method using a WSD module is presented. It is based on *maximum entropy conditional probability models.* The method presented consists of three sub-tasks. First of all, the process of obtaining the semantic role needs the sense of the target verb. After that, several heuristics are applied in order to obtain the arguments of the sentence. And finally, the semantic roles that fill these arguments are obtained. Training and development data are used to build this learning system.

Results about the VSD, ABD and SRD modules have been shown. Currently, we are working on the definition of new features to the SRD modules. So, the re-definition of the heuristics is planned in order to improve the results. After that, we are going to work on the tuning phase in order to achieve an optimum identification of the semantic roles.

On the other hand, we are working in the integration of semantic aspects in Question Answering or Information Retrieval Systems, in order to obtain High Precision QA/IR Systems. Shortly, we will show results about this incorporations.

References

1. *Eighth Conference on Natural Language Learning (CoNLL-2004),* Boston, MA, USA, Mayo 2004.
2. U. Baldewein, K. Erk, S. Padó, and D. Prescher. Semantic role labeling with chunk sequences. In *Proceedings of the Eighth Conference on Natural Language Learning (CoNLL-2004)* [1].
3. X. Carreras and L. Màrquez. Introduction to the CoNLL-2004 Shared Task: Semantic Role Labeling. In *Proceedings of the Eighth Conference on Natural Language Learning (CoNLL-2004)* [1].
4. X. Carreras, L. Màrquez, and G. Chrupala. Hierarchical Recognition of Propositional Arguments with Perceptrons. In *Proceedings of the Eighth Conference on Natural Language Learning (CoNLL-2004)* [1].
5. M. Fleischman, N. Kwon, and E. Hovy. Maximum Entropy Models for FrameNet Classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLPS003),* July 2003.
6. J. Giménez and L. Màrquez. Fast and Accurate Part-of-Speech Tagging: The SVM Approach Revisited. In *Proceedings of Recent Advances in Natural Language Processing 2003,* Borovets, Bulgaria, Septiembre 2003.
7. K. Hacioglu, S. Pradhan, W. Ward, J.H. Martin, and D. Jurafsky. Semantic Role Labeling by Tagging Syntactic Chunks. In *Proceedings of the Eighth Conference on Natural Language Learning (CoNLL-2004)* [1].
8. D. Higgins. A transformation-based approach to argument labeling. In *Proceedings of the Eighth Conference on Natural Language Learning (CoNLL-2004)* [1].
9. B. Kouchnir. A Memory-based Approach for Semantic Role Labeling. In *Proceedings of the Eighth Conference on Natural Language Learning (CoNLL-2004)* [1].
10. J. Lim, Y. Hwang, S. Park, and H. Rim. Semantic role labeling using maximum entropy model. In *Proceedings of the Eighth Conference on Natural Language Learning (CoNLL-2004)* [1].
11. M.P. Marcus, B. Santorini, and M.A. Marcinkiewicz. Building a large annotated corpus of english: the penn treebank. *Computational Linguistics,* (19), 1993.

12. P. Moreda, M. Palomar, and A. Suárez. Identifying semantic roles using maximum entropy models. In *Proceedings of the International Conference Text Speech and Dialogue,* Lecture Notes in Artificial Intelligence, Brno, Czech Republic, 2004. Springer-Verlag.
13. M. Palmer, D. Gildea, and P. Kingsbury. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics,* 2004. Submitted.
14. K. Park, Y. Hwang, and H. Rim. Two-Phase Semantic Role Labeling bsed on Support Vector Machines. In *Proceedings of the Eighth Conference on Natural Language Learning (CoNLL-2004)* [1].
15. V. Punyakanok, D. Roth, W. Yih, D. Zimak, and Y. Tu. Semantic Role Labeling Via Generalized Inference Over Classifiers. In *Proceedings of the Eighth Conference on Natural Language Learning (CoNLL-2004)* [1].
16. A. Suárez and M. Palomar. A maximum entropy-based word sense disambiguation system. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING),* pages 960–966, Taipei, Taiwan, Agosto 2002.
17. A. van den Bosch, S. Canisius, I. Hendricks, W. Daelemans, and E.T.K. Sang. Memory-based semantic role labeling: Optimizing features, algorithm and output. In *Proceedings of the Eighth Conference on Natural Language Learning (CoNLL-2004)* [1].
18. K. Williams, C. Dozier, and A. McCulloh. Learning Transformation Rules for Semantic Role Labeling. In *Proceedings of the Eighth Conference on Natural Language Learning (CoNLL-2004)* [1].

Multi-session Management in Spoken Dialogue System

Hoá Nguyen and Jean Caelen

University of Joseph Fourier, Laboratory CLIPS,
BP 53, 38041 Grenoble Cedex 9, France
{Ngoc-Hoa.Nguyen, Jean.Caelen}@imag.fr
http://www-clips.imag.fr/

Abstract. In order to increase the role of machines in supporting more capabilities as regards a spoken dialogue system, we present in this paper a new problem incorporating multi-session in such a system. Instead of only handling single dialogue, such a system can take an intermediary role to communicate with many users in several discontinuous sessions for reaching a compromise between them. We describe here a new approach for modeling the multi-session and then we concentrate on the multi-session management of such a system dedicated for a complete service having several tasks.

1 Introduction

The spoken dialogue system has attracted much attention as the way of communicating with machines through speech. These systems normally enable users to interact with them and to perform a certain task. For example the CMU Communicator system is aimed at helping a user to create a travel itinerary consisting of air transportation, hotel reservations and car rentals [13], ARISE allows the users to consult the train timetable [2], TRINDI enables the users to make choices in the performance of the route planning [14], etc. The dialogue in these systems in particular, and more generally in the actual dialogue systems, contains just some exchanges between a user and the system.

In the context of company's voice portal PVE (Portail Vocal d'Entreprise) project [15], our analysis of use, which we carried out in hospitals, judicial and company offices, show that the voice service is naturally very useful for applications such as information requests, confirmation of a request, secretarial work such as transferring calls, scheduling appointments, reserving rooms... Spoken dialogue in these situations is normally short but contains complex utterances. However, users always require a complete service that is defined like a complete resolving problem in a face-to-face situation. For example, in the room reservation service, the spoken dialogue system must act as, behave as and take on the role of a virtual secretary. This means that the user is not only able to reserve a room, but also to request the confirmation of all participants and their availability. Moreover, the user should also be able to ask the system to negotiate with others in order to obtain a good compromise between them.

Let see the following example: One user D would like to book the Lafayette room and he calls the system S. Unfortunately, this room is already taken by the person P. However, D has greater priority than P (may be due to hierarchical position), so he asks the system to contact, to tell P to leave this room for him. The system then contacts P and fortunately reaches an agreement with him: he accepts to put back his meeting to the next day. Once the system has the response, it will recall D to inform him about the results.

S1: Person D + System S

D: hello, I am D, could you book me the room Lafayette for tomorrow at 9 o' clock, please?
S: I'm sorry Mr. D, this room is already taken by Mr. P...
D: Tell him I need it and could he leave this room for me.
S: OK, I'll contact him and I'll keep you up to date.

S2: System S + Person P

S: hello, are you Mr. P?
P: yes,
S: I'm contacting you about the Lafayette reservation. Could you leave this room for Mr. D, please?
P: Let me see... OK, I'll put back my meeting to tomorrow.
S: That's great, thank you very much.

S3: System S + Person D

S: Hello, Mr. D?
D: Yes, it's me
S: Mr. P has already agreed to leave the Lafayette for you at 9 o'clock tomorrow.
D: That's very nice, thank you.

Thus, we can see that the users require more functionality towards a dialogue system: the spoken dialogue system should now take the role of a mediator to negotiate with several users in order to resolve the conflict between them. There are possibly multiple users engaged in a dialogue. Therefore, we consider now the dialogue is expressed by multi-session with multiple users; each session is a dialogue between a single user and the system. In this paper, we introduce an approach for modeling the multi-session dialogue, and then the mechanisms to manage them in a spoken dialogue system.

2 Basic Principles

This section describes some important elements, which are used for our multi-session modeling. In relation to the architecture for a spoken dialogue system, we used the modular/multi-agent architecture described in [8] and as illustrated in figure 1. The multi-session management shown in the session 4 will be implemented in two modules: the "dialogue manager" and the "task manager".

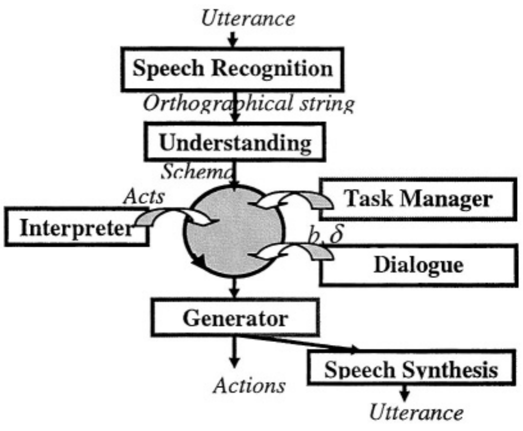


Fig. 1. Architecture for a Spoken Dialogue System

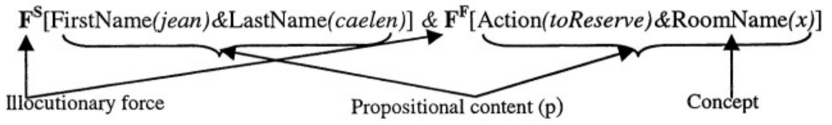
2.1 Speech Act

Austin [1] and Searle [11] consider all utterance as an act of communication called a speech act. A speech act might contain just one word, several words or a complete sentence. By combining with the notion of illocutionary logic, Vanderveken [12] defined the illocutionary force of a speech act. Then, as Caelen [3], it is useful to retain the following illocutionary forces in the human-machine dialogue domain:

Table 1. Illocutionary Forces of a Dialogue Act

Act	Signification
F^A	Do or execute an action.
F^F	Ask the hearer to perform an action.
F^S	Communicate information in assertive way.
F^{FS}	Ask for information.
F^P	Give a choice, make an invite.
F^D	Oblige to do without giving an alternative.

Based on speech act theory and illocutionary logic, we define the notion of a dialogue act. A dialogue act is a speech act that is annotated by the illocutionary force. We represent a dialogue act as an illocutionary force that specifies what the speaker wishes to achieve, and a propositional content representing the semantic schema of statement. Each utterance can contain more than one dialogue act. For example, the utterance "Jean Caelen is calling... I would like to book a conference room" may be interpreted as follows:


F^S[FirstName(jean)&LastName(caelen)] & F^F[Action(toReserve)&RoomName(x)]
Illocutionary force
Propositional content (p)
Concept

The user dialogue act is built naturally from user's utterance by the *Interpreter* module and the dialogue manager has to generate the system dialogue act as his response to user's utterance.

2.2 Dialogue Goal

A goal is generally a task state or a mental state that one wishes to reach (for example: to obtain information, to resolve a problem, etc.). The start of an exchange (a series of talking turns during which a goal is sustained) is initiated by the emergence of a new goal. Then this goal is transformed during the exchange and becomes a final goal (a task state or of a situation at the end of an exchange) at which point the exchange ends by a success or by a failure. The success obeys the double condition of being a reached goal and a satisfied goal. The final goal is not always predictable at the start.

A dialogue goal is the goal that is sustained during an exchange. In the human-machine, dialogue it results from the type of considered task. For example, a room reservation implies a goal (of the task) as a request for a room and a dialogue goal that leads to a communication/negotiation to reach the goal. Thus, the dialogue goal can be satisfied while a general goal may not necessarily be satisfied [4]. A dialogue goal is represented as a logical predicat **b** and its possible states are shown in the following table:

Table 2. Evolution of the State of Dialogue Goal

Symbol	Status	Description
?b	new	This goal has just been expressed by user.
†b	reached	The predicate b becomes true.
‡b	satisfied	User manifests their agreement on †b, this agreement can be explicit or implicit.
-b	pending	System solves temporarily another problem.
b'	repaired	Due to a lack of understanding, the goal is modified; user does not go back on his previous goal.
sb	sub-goal	The problem is decomposed into sub-problems.
@b	abandoned	This state is result of a failure or a voluntary abort.

A dialogue goal is formed by the abstraction of dialogue act helped by the dialogue plan (which is specified in the task model by elementary goals, called task goals, and managed by the task manager). Once the dialogue manager has formed the dialogue goal, it sends this goal to the task manager to know if this goal is either reached, unable to reach, or missed information (states related to tasks). And then, the dialogue manager must decide itself if this dialogue goal is satisfied, pending or left [5].

2.3 Dialogue Strategy

The dialogue strategy **δ** is the way to handle the talking turns between a user and the system to lead the final state (satisfied or abandoned) of the dialogue goal. The strategy aims at choosing the best adjustment direction of the goals at a given moment. It is strongly a decisive factor in the dialogue efficiency, which is calculated by the speed of convergence of the dialogue acts towards the final goal. We distinguish the types of dialogue strategy by two different categories as following [5]:

Non-inferential Strategies: the strategies that the system does not need to know initially the user's goal:

- Directive strategy: consists in keeping the initiative with the system to drive the dialogue, maintaining the exchange goal and, imposing a new goal.
- Reactive strategy: is used to delegate the initiative to the user either by making him endorse the dialogue goal, or by adopting this goal.
- Constructive strategy: consists in moving the current goal in order to invoke a detour, for example to make it notice an error, make a quotation, and undo an old fact...

Inferential Strategies: These strategies are said to be inferential when both user and system need a perceptive knowledge of their respective goals. In these strategies, the two speakers have a shared initiative:

- Cooperative strategy: consists in adopting the goal of the user by proposing one or many solutions which directly bring the best way to reach his goal.
- Negotiated strategy: can be involved in a situation where the goals are incompatible and the both user and system want to minimize the concessions. The negotiation is expressed by argumentative sequences (argumentation/refutation) with the proposal for a sub-optimal solution until convergence or acknowledgement of failure.

3 Multi-session Modeling

Suppose now that a spoken dialogue system must perform a complete service having several tasks. We consider that a dialogue initiated by a user D for satisfying a goal related to this service is divided into a set of discontinuous sub-dialogues, each sub-dialogue representing a session S_k, including an opening phase, the different speech turns between the concerned user P_k and the system S, and a closure phase. Therefore, the framework of dialogue is a sequence of sessions, the first one with the requester D, then and the next with the different addressees P_k if necessary and at the end with D for the conclusion.

3.1 Definitions

We suppose during the first session S_1, the user D interacts with the system S for resolving the goal b^D. There are three possible cases at the closure of the session:

1. b^D is satisfied (noted by $\ddagger b^D$),
2. D chooses to abort his goal (noted by $@b^D$),
3. b^D cannot be reached because it is in conflict with others goals previously satisfied by others users of the service.

The third case leads new sessions to try to resolve b^D: in a first step the system S put the goal b^D in the pending state (noted $-b^D$) and then expands the different solutions to resolve the conflict and initiates a negotiation with the users which goals are in conflict with b^D.

We define:

- *Dialogue goal in conflict* b_f: the dialogue goal animated by the requester D is in conflict with the one already satisfied by the user P: $b_f = (-b^D, \ddagger b^P)$
- *Tree of dialogue goals in conflict:* more generally the goal b^D is possibly in conflict with the n satisfied goals of m other users, called for the next the *'patients'*, $(P_1,\ldots P_m)$ related by AND/OR operators. This set of conflict goals $T_f=(b_{f1}, b_{f2}, \ldots , b_{fm})$ makes a AND-OR tree of dialogue goals in conflict with b^D. Each leaf of this tree represents a goal in conflict from the patient P_k.

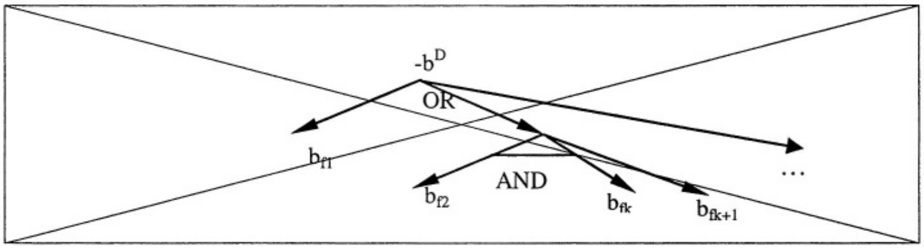


Fig. 2. A Tree T_f of Dialogue Goals in Conflict

The resolution of the conflict for b^D is to find a path from leafs to the root in respect to the AND/OR conditions along the tree. The resolution of one particular goal in conflict b_{fk} in a leaf should be done by a special session issued of a dialogue S_k with the user P_k.

3.2 Session Coordination

Thus, the resolution of b^D needs to manage dynamically several new sessions. The sessions sequence obeys the exploration of the AND-OR tree T_f. The goal b^D will be satisfied if and only if there is at least a path in T_f having all satisfied elements.

The algorithm to reach b_D during sessions is shown as following:

```
While $b^D$ is not reached Do,
    From the tree $T_f$ Extract the best path unmarked [7] to reach $b^D$ and For all the leaf along this path, Open a negotiation dialogue with the concerned patient in order to solve the local conflict,
    In case of breakdown Mark the path and Try again from the previous step
    In case of success Stop
EndWhile
Notify the result to D: $b^D$ is reached (noted by $\dagger b^D$) or abandoned ($@b^D$). From here, D could of course accept or not this result and then the dialogue could continue
```

In the negotiation process, the system performs each session separately, but the order of handling these sessions depends on the best path founded at each step.

4 Multi-session Management

The multi-session management has to be done through both the dialogue manager and the task manager. The main idea here is how to manage efficiently the tree of dialogue goal in conflict. As our approach, the dialogue manager will control both the dialogue goal in a session and the tree of dialogue goals in conflict $\mathbf{T_f}$. In relation to the task manager, it will control the triggering, the development/execution of a session, and moreover, the coordination of the sessions sequence.

4.1 At the Dialogue Manager Level

In this section, we are only interested in the management of the goal in conflict (the management of a normal dialogue goal as well as the dialogue strategy were described in [9][6], and we do not mention them here). The task manager should compute and send the tree of dialogue goals in conflict $\mathbf{T_F}$ to the dialogue manager. Once the dialogue manager receives the tree it manages the sessions and interacts with the task manager which acts as a problem-planner.

During each negotiation, the goal b_{fi} goes forward according to the attitudes of $\mathbf{P_i}$ towards b_{fi}.Its possible attitudes are:

- give up b_{fi} to D without conditions,
- do not abandon b_{fi} in all cases,
- leave out b_{fi} to D within conditions as modifying b_{fi}, requesting a new goal b'_{fi}.

In the two first cases, it seems that are not complicated as the third, which depends on new conditions of $\mathbf{P_i}$, which can be:

- feasible without the influence of others P,
- not feasible,
- feasible but it can lead to a new conflict with another P via a new session.

These attitudes manifest directly to b_f via the dialogue acts of user and are recognized by the dialogue manager. The negotiation process for b^D finishes when it has been reached $\dagger b^D$, or all of possible negotiations have been failed and D has to abandon his goal $@b^D$.

4.2 At the Task Manager Level

The task manager clearly takes an important role in relation to the multi-session management in a spoken dialogue system. For ensuring the coherence of multi-sessions, it should contain the planning of all possible sessions, manage the multi-session sequence, and supervise progress of the goal in conflict.

During a session, the task manager must dynamically build $\mathbf{T_f}$ in case of having conflicts. Once user requests to perform T_f, the task manager will develop a plan to negotiate with patients. For each patient P_i, the task manger will launch a session to resolve b_{fi}, and once the dialogue manager has the b^D state that has already reached or abandoned.

5 Example

For modeling the multi-session in a spoken dialogue system, we used the room's reservation service via telephone as a case study. Let us use the above example in section 1 to illustrate our approach:

In S1, the requester D manifests directly a new goal $?b^D$ = **person(D)** $\wedge$ **room(R)** $\wedge$ **date(DT)** $\wedge$ **toReserve(D,R,DT)**. However, the room requested by D was already reserved by P as $‡b^P$ = **person(P)** $\wedge$ **room(R)** $\wedge$ **date(DT)** $\wedge$ **toReserve(P,R,DT)**.

By interacting with the task manager, the dialogue manager determines a room and date conflict represented by $b_f = (b^D, ‡b^P)$. The task manager then creates $T_f = \{b_f\}$ and the dialogue manager plans a new session to negotiate with P.

In the next step, the task manager interacts with the dialogue manager to launch a new session S_2 for resolving b_f. The system S calls P and suppose the negotiation in this case happens successfully: P accepts for moving his meeting to the next day so the goal in conflict has been resolved, because $‡b^P$ becomes $‡b'^P$ = **person(P)** $\wedge$ **room(R)** $\wedge$ **date(DT+1)** $\wedge$ **toReserve(P,R,DT+1)**. The task manager should acknowledge these new situations and plans making a new session to inform D about the results.

The third session S_3 is just to notify to D the state of T_f, a reached goal now. Naturally, D could also deny b_f by such reasons, but fortunately, he recognizes b_f and manifests it to be satisfied. So the dialogue animated by D has been completed.

6 Results and Conclusion

Multi-session management in a discontinuous human-machine dialogue has become necessary in increasing the capability of the spoken dialogue system. Based on the dialogue management which is reduced as much as possible the dependence on task model, we have built a prototype of such a system dedicated for the reservation service aimed in the PVE project (by French language). Our prototype could currently manipulate the sessions like the room reservation, meeting convocation, and moreover, the cancellation/modification of a reservation. By applying our methodology of multi-session modeling and management, our prototype can now act like a real mediator: users could ask the system to negotiate with another user in case having conflicts of room, date.

The experimentations, which have carried out with our prototype with the corpus collected during the Wizard of Oz step in the PVE project, prove the validity of our theory for the multi-session management. We have also done a lot of tests within multi-session for resolving the room/date conflict, and we will publish the official result evaluation later In the near future, with the speech-recognized improvement, the robust comprehension/interpretation, our system will be totally completed with the best negotiation capability.

The first results we have obtained and are obtaining not only show the importance of multi-session management in a spoken dialogue system, but also open a new direction in the way of bringing intelligence and speech to machines.

References

1. Austin, J. L., *How to Do Things with Words,* Harvard University Press, 1962.
2. Baggia P., Gauvain J.L., Kellner A., Perennou G., Popovici C., Sturm J., Wessel F., "Language Modelling and Spoken Dialogue Systems - the ARISE experience", in Proc. Sixth European Conference on Speech Communication and Technology, September 1999.
3. Caelen, J., "Attitudes cognitives et actes de langage". In Du dialogue, Recherches sur la philosophie du langage, n° 14, Vrin éd., Paris, p. 19-48, 1992.
4. Caelen J., "Modèles formels de dialogue", Actes du GdR I3 - *Information, Interaction Intelligence,* CEPADUES Editions, p. 31-58, 2002.
5. Caelen J., Stratégies de dialogue, Actes de conférence MFI'03 (Modèles Formels de I'Interaction), Lille, CEPADUES ed, 2003
6. Caelen J., Nguyen H., "Gestion de buts de dialogue", the 11th conference on Natural Language Processing - TALN, Morocco, April 2004.
7. Mittal S., Dyn C.L., Morjaria M.. "PRIDE: An expert system for the design paper handling systems", in Application of knowledge-based systems to engineering analysis and design, C.L. Dyn ed., American Society of Mechanical Engineers, 1985
8. Nguyen H., "Vers une architecture générique de système de dialogue oral homme-machine". RECITAL-France 6-2003.
9. Nguyen H., Caelen J., "Generic manager for spoken dialogue systems". DiaBruck: 7th Workshop on the Semantics and Pragmatics of Dialogue, Proceedings pp.201-202, 2003.
10. Peckham K., *A new generation of spoken dialogue systems: Results and Lessons from the SUNDIAL project,* Proceedings of Eurospeech, p. 33-40, 1993.
11. Searle, J., *Speech Acts: An Essay in the Philosophy of Language,* Cambridge University Press, 1969.
12. Vanderveken D., *Meaning and Speech Acts,* Volumes 1 and 2, Cambridge University Press, 1990.
13. Xu, W. and Rudnicky, A. "Task-based dialog management using an agenda". ANLP/NAACL 2000 Workshop on Conversational Systems, May 2000, pp. 42-47.
14. http://www.ling.gu.se/projekt/trindi/
15. www.telecom.gouv.fr/rnrt/projets/res_01_5.htm

Semantically-Driven Explanatory Text Mining: Beyond Keywords*

John Atkinson-Abutridy

Departamento de Ingeniería Informática,
Universidad de Concepción, Chile
atkinson@inf.udec.cl

Abstract. In this paper, a new explanatory and high-level approach to knowledge discovery from texts is described which uses natural language techniques and and evolutionary computation based optimization to find novel patterns in textual information. In addition, some results showing the promise of the approach towards effective text mining when compared to human performance are briefly discussed.

1 Introduction

Knowledge Discovery from Texts (KDT) or Text Mining, is an emerging technology for analysing large collections of unstructured documents for the purposes of extracting interesting and novel knowledge [8], so it can be seen as a leap from Data Mining and Knowledge Discovery from Databases (KDD).

However, Data Mining techniques cannot be immediately applied to text data for the purposes of TM as they assume a structure in the source data which is not present in free text. In addition, while the assessment of discovered knowledge in the context of KDD is a key aspect for producing an effective outcome, the assessment of the patterns discovered from text has been a neglected topic in the majority of the text mining approaches and applications.

One of the main features of most sophisticated approaches to text mining which use high-level representation (i.e., not just keywords) is an intensive use of electronic linguistic resources including ontologies, thesauri, etc., which highly restricts the application of the unseen patterns to be discovered, and their domain independence.

In this context, using evolutionary computation techniques (i.e., Genetic Algorithms) for mining purposes [5] has several promising advantages over the usual analysis methods employed in KDT: the ability to perform global search, the exploration of solutions in parallel, the robustness to cope with noisy and

* This research is sponsored by the National Council for Scientific and Technological Research (FONDECYT, Chile) under grant number 1040469 *"Un Modelo Evolucionario de Descubrimiento de Conocimiento Explicativo desde Textos con Base Semántica con Implicaciones para el Análisis de Inteligencia."*

missing data, and the ability to assess the goodness of the solutions as they are produced in terms of the problem domain.

Accordingly, in this paper we discuss a new semantically-guided model for KDT which brings together the benefits of shallow text processing and multi-objective evolutionary computation to produce novel and interesting knowledge.

2 Text Mining and Knowledge Discovery from Texts

In performing typical text mining tasks, many researchers worldwide have assumed a typical Bag-of-Words (BOW) representation for text documents which makes it easy to analyse them but restrict the kind of discovered knowledge. Furthermore, the discoveries rely on patterns in the form of numerical associations between terms from the documents which fail to provide explanations of, for example, why these terms show a strong connection. In consequence, no deeper knowledge or evaluation of the discovered knowledge are considered.

Many recent KDT and text mining applications show a tendency to start using more structured or deeper representations than just keywords (or terms) to perform further analysis so to discover unseen patterns. Early research on this kind of view can be due to seminal work by Swanson [10] on exploratory analysis from the titles of articles stored in the MEDLINE medical database. Swanson designed a system to infer key information by using simple patterns which recognize causal inferences such as "`X cause Y`" and more complex implications, which lead to the discovery of hidden and previously neglected connections between concepts. This work provided evidence that it is possible to derive new patterns from a combination of text fragments plus the explorer's medical expertise.

Further approaches have exploited these ideas by combining more elaborated *Information Extraction* (IE) patterns and general lexical resources (e.g., WordNet) [6] or specific concept resources. They deal with automatic discovery of new lexicosemantic relations by searching for corresponding defined patterns in unrestricted text collections so as to extend the structure of the given ontology.

A different view in which linguistic resources such as WordNet are used to assist the discovery and to evaluate the unseen patterns is followed by Mooney and colleagues [1] who propose a system to extract basic information (i.e., rules containing attribute-value pairs) from general documents by using IE extraction patterns. Furthermore, human subject assess the real interestingness of the most relevant patterns mined by the system. The WordNet approach to evaluation has proved to be well correlated with human judgments, and unlike the other methods, Mooney's is the only one which deals with the whole process of knowledge discovery: mining and patterns evaluation. However, the dependence on the existence of an linguistic resource prevents the method from dealing with specific terminology leading to missing and/or misleading information.

3 A LSA-Guided Model for Effective Text Mining

We developed a semantically-guided model for evolutionary Text Mining which is domain-independent but genre-based. Unlike previous approaches to KDT, our approach does not rely on external resources or descriptions hence its domain-independence. Instead, it performs the discovery only using information from the original corpus of text documents and from the training data generated from them. In addition, a number of strategies have been developed for automatically evaluating the quality of the hypotheses.

We have adopted Genetic Algorithms (GAs) as central to our approach to KDT. However, for proper GA-based KDT there are important issues to be addressed including representation and guided operations to ensure that the produced offspring are semantically coherent.

In order to deal with these issues so to produce an effective text mining process, our working model has been divided into two phases. The first phase is the preprocessing step aimed to produce both training information for further evaluation and the initial population of the GA. The second phase constitutes the knowledge discovery itself, in particular this aims at producing and evaluating explanatory unseen hypotheses.

The whole processing starts by performing the IE task which applies extraction patterns and then generates a rule-like representation for each document of the specific domain corpus. Once generated, these rules, along with other training data, become the "model" which will guide the GA-based discovery.

In order to generate an initial set of hypotheses, an initial population is created by building random hypotheses from the initial rules, that is, hypotheses containing predicate and rhetorical information from the rules are constructed. The GA then runs for a number of generations until a fixed number of generations is achieved. At the end, a small set of the best hypotheses (i.e., discovered patterns) are obtained.

The description of the approach is organized as follows: section 3.1 presents the main features of the text preprocessing phase and how the representation for the hypotheses is generated. In addition, training tasks which generate the initial knowledge (semantic and rhetorical information) to feed the discovery are described. Section 3.2 describes constrained genetic operations to enable the hypotheses discovery, and proposes different evaluation metrics to assess the plausibility of the discovered hypotheses in a multi-objective context.

3.1 Text Preprocessing and Representation

The preprocessing phase has two main goals: to extract important information from the texts and to use that information to generate both training data and the initial population for the GA.

It is well-known that processing full documents has inherent complexities [9], so we have restricted our scope somewhat to consider a scientific genre involving scientific/technical abstracts. From this kind of abstract's structure, important constituents can be identified such as *Rhetorical Roles, Predicate Relations,* and *Causal Relation(s).*

In order to extract this initial key information from the texts, an IE module was built. Essentially, it takes a set of text documents, has them tagged through a previously trained Part-of-Speech (POS) tagger, and produces an intermediate representation for every document which is then converted into a general rule.

In addition, key training data are captured from the corpus of documents itself and from the semantic information contained in the rules. This can guide the discovery process in making further similarity judgments and assessing the plausibility of the produced hypotheses.

Following work by [7] on *Latent Semantic Analysis* (LSA) incorporating structure, we have designed a semi-structured LSA representation for text data in which we represent predicate information and arguments separately once they have been properly extracted in the IE phase.

We also included basic knowledge at a rhetorical, semantic level, and co-occurrence information which can be effectively computed to feed and guide the evolutionary discovery process. Accordingly, we perform two kinds of tasks: creating the initial population and computing training information from the rules (i.e., correlations between rhetorical roles and predicate relations, co-occurrences of rhetorical information, etc).

3.2 Evolutionary Text Mining and Patterns Evaluation

Our model for evolutionary Text Mining is strongly guided by semantic and rhetorical information, and consequently there are some soft constraints to be met before producing the offspring so as to keep them coherent.

The discovery process itself (i.e., multi-objective Genetic Algorithm) will start from a initial population, which in this case, is a set of semi-random hypotheses built up from the preprocessing phase. Next, constrained GA operations are applied and the hypotheses are evaluated. In order for every individual to have a fitness assigned, we use a evolutionary multi-objective optimization strategy based on the SPEA algorithm [11] in a way which allows incremental construction of a Pareto-optimal set and uses a steady-state strategy for the population update.

Pattern Discovery. Using the semantic measure previously highlighted and additional constraints discussed later on, we propose new operations to allow guided discovery such that unrelated new text knowledge is avoided, as follows:

- *Selection:* selects a small number of the best parent hypotheses of every generation (*Generation Gap*) according to their Pareto-based fitness.
- *Crossover:* a simple recombination of both hypotheses' conditions and conclusions takes place, where two individuals swap their conditions to produce new offspring (the conclusions remain).

  Under normal circumstances, crossover works on random parents and positions where their parts should be exchanged. However, in our case this operation must be restricted to preserve semantic coherence. We use soft semantic constraints to define two kind of recombinations:

1. *Swanson's Crossover:* based on Swanson's hypothesis [10] we propose a recombination operation as follows:
   If there is a hypothesis (AB) such that "IF A THEN B" and another one (BC) such that "IF B' THEN C", (B' being something semantically similar to B) then a new interesting hypothesis "IF A THEN C" can be inferred, only if the conclusions of AB have high semantic similarity (i.e., via LSA) with the conditions of hypothesis BC.
2. *Default Semantic Crossover:* if the previous transitivity does not apply then the recombination is performed as long as both hypotheses as a whole have high semantic similarity which is defined in advance by providing minimum thresholds.

– *Mutation:* aims to make small random changes on hypotheses to explore new possibilities in the search space. This is performed in a semantically-constrained way at roles, arguments, and predicate levels.
– *Population Update:* we use a non-generational GA in which some individuals are replaced by the new offspring in order to preserve the hypotheses' good material from one generation to other, and so to encourage the improvement of the population's quality. We use a steady-state strategy in which each individual from a small number of the worst hypotheses is replaced by an individual from the offspring only if the latter are better than the former.

Automatic Evaluation of Discovered Patterns. Since each pattern (hypothesis) discovered by the model has to be assessed by different criteria, usual methods for evaluating fitness are not appropriate. Hence *Evolutionary Multi-Objective Optimization* (EMOO) techniques which use the multiple criteria defined for the hypotheses are needed. For this, we propose EMOO-based evaluation metrics to assess the hypotheses' fitness in a domain-independent way.

In order to establish evaluation criteria, we have taken into account different issues concerning plausibility, and quality itself. Accordingly, we have defined eight evaluation criteria to assess the hypotheses given by: **relevance, structure, cohesion, interestingness, coherence, coverage, simplicity, plausibility of origin**.

Evaluation criteria by which the hypotheses are assessed and the questions they are trying to address are as follows:

– **Relevance** *(How important is the hypothesis to the target question?):* measures the semantic closeness between the hypothesis' predicates (relations and arguments) and the target concepts. Relevance is then computed from compound vectors obtained in the LSA analysis which follows work by Kintsch on *Predication* [7]. We then propose an adaptation of the LSA-based closeness so to compute the overall relevance of the hypothesis in terms of the "strength" which determines how closely related two concepts are to both some predicate and its arguments.

- **Structure** *(How good is the structure of the rhetorical roles?):* measures how much of the rules' structure is exhibited in the current hypothesis. Since we have previous preprocessing information regarding bi-grams of roles, the structure is computed by following a Markov chain of the "bi-grams" of the rhetorical information of each hypothesis.
- **Cohesion** *(How likely is a predicate action to be associated with some specific rhetorical role?):* measures the degree of "connection" between rhetorical information and predicate actions. The issue here is how likely some predicate relation in the current hypothesis is to be associated with some rhetorical role.
- **Interestingness** *(How interesting is the hypothesis in terms of its antecedent and consequent?):* Unlike other approaches to measure "interestingness" which use an external resource and rely on its organisation we propose a different view where the criterion can be evaluated from the semi-structured information provided by the LSA analysis. The measure for a hypothesis is defined as a degree of semantic dissimilarity (unexpectedness) between its antecedent and its consequent.
- **Coherence:** This metrics addresses the question whether the elements of the current hypothesis relate to each other in a semantically coherent way (*text coherence* [3,4]). We developed a simple method to measure coherence, following work by [4] on measuring text coherence. Semantic coherence is calculated by considering the average semantic similarity between consecutive elements of the hypothesis.
- **Coverage:** The coverage metric tries to address the question of how much the hypothesis is supported by the model (i.e., rules representing documents and semantic information).
  In order to deal with the criterion in the context of text mining, we say that a hypothesis covers an extracted rule only if the predicates of this hypothesis are roughly (or exactly, in the best case) contained in that rule.
- **Simplicity** *(How simple is the hypothesis?):* shorter and/or easy-to-interpret hypotheses are preferred. Since the criterion has to be maximized, the evaluation will depend on the length (number of elements) of the hypothesis.
- **Plausibility of Origin** *(How plausible is the hypothesis produced by Swanson's evidence?):* If the current hypothesis was an offspring from parents which were recombined by a Swanson's transitivity-like operator, then the higher the semantic similarity between one parent's consequent and the other parent's antecedent, the more precise is the evidence, and consequently worth exploring as a novel hypothesis.

Note that since we are dealing with a multi-objective problem, there is no simple way to get independent fitness values as the fitness involves a set of objective functions to be assessed for every individual. Therefore the computation is performed by comparing objectives of one individual with others in terms of *Pareto dominance* [2] in which non-dominated solutions are searched for in every generation.

We took a simple approach in which an approximation to the Pareto optimal set is incrementally built as the GA goes on. The basic idea is to determine whether a solution is better than other in global terms, that is, a child is better if this is a candidate to become part of the Pareto front.

Next, since our model is based on a multi-criteria approach, we have to face three important issues in order to assess every hypothesis' fitness: Pareto dominance, fitness assignment and the diversity problem [2]. Despite an important number of state-of-the-art methods to handle these issues [2], only a small number of them has focused on the problem in an integrated and representation-independent way. In particular, Zitzler [11] proposes an interesting method, *Strength Pareto Evolutionary Algorithm* (SPEA) which uses a mixture of established methods and new techniques in order to find multiple Pareto-optimal solutions in parallel, and at the same time to keep the population as diverse as possible. We have also adapted the original SPEA algorithm which uses an elitist strategy to allow for the incremental updating of the Pareto-optimal set along with our steady-state replacement method.

4 Experiments and Results

In order to assess the quality of the discovered knowledge (hypotheses) by the model, a computer prototype has been built. For the purpose of the experiments, the corpus of input documents has been obtained from agricultural articles. Next, a set of 1000 documents was extracted from which one third were used for setting parameters and making general adjustments, and the rest were used for the GA itself in the evaluation stage.

Next, we tried to provide answers to two basic questions concerning our original aims: How well does the GA for text mining behave? and How good are the hypotheses produced according to human experts in terms of text mining's ultimate goals: interestingness, novelty and usefulness, etc.

In order to address these issues, we used a methodology consisting of two phases: the system evaluation and the experts' assessment.

1. *System Evaluation:* this aims at investigating the behavior and the results produced by the GA.

   We set the GA by generating an initial population of 100 semi-random hypotheses. In addition, we defined the main global parameters such as *Mutation Probability* (0.2), *Crossover Probability* (0.8), *Maximum Size of Pareto set* (5%), etc. We ran five versions of the GA with the same configuration of parameters but different pairs of terms to address the quest for explanatory novel hypotheses.

   The different results obtained from running the GA as used for our experiment are shown in the form of a representative behavior in figure 1, where the number of generations is placed against the average objective value for some of the eight criteria.

   Some interesting facts can be noted. Almost all the criteria seem to stabilize after generation 700 for all the runs, that is, no further improvement beyond this point is achieved and so this may give us an approximate indication of the limits of the objective function values.

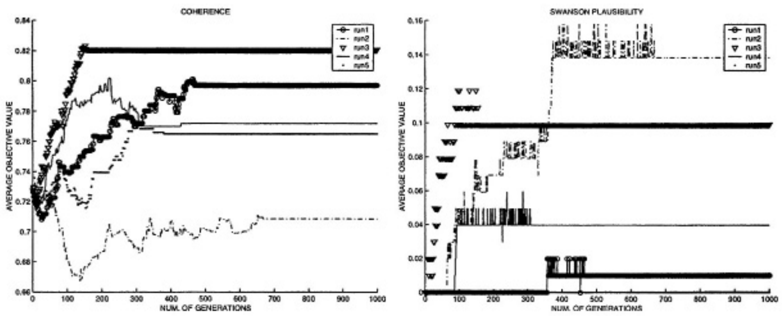


Fig. 1. Evolutionary Evaluation for some of the criteria

Note also that not all the criteria move in the same direction. Look at the results for the criteria for the same period of time, between generations 200 and 300 for run 4. For an average hypothesis, the quality of *Coherence, Cohesion, Simplicity* and *Structure* gets worse, whereas this improves for *Coverage, Interestingness* and *Relevance,* and has some variations for (Swanson) *Plausibility.*

2. *Expert Assessment:* this aims at assessing the quality (and therefore, effectiveness) of the discovered knowledge on different criteria by human domain experts. For this, we designed an experiment in which 20 human experts were involved and each assessed 5 hypotheses selected from the Pareto set. We then asked the experts to assess the hypotheses from 1 (worst) to 5 (best) in terms of the following criteria: Interestingness (INT), Novelty (NOV), Usefulness (USE), Sensibleness (SEN), etc.

In order to select worthwhile terms for the experiment, we asked one domain expert to filter pairs of target terms previously related according to traditional clustering analysis. The pairs which finally deserved attention were used as input in the actual experiments (i.e., **glycocide and inhibitors**).

Once the system hypotheses were produced, the experts were asked to score them according to the five subjective criteria. Next, we calculated the scores for every criterion as seen in the overall results in figure 2.

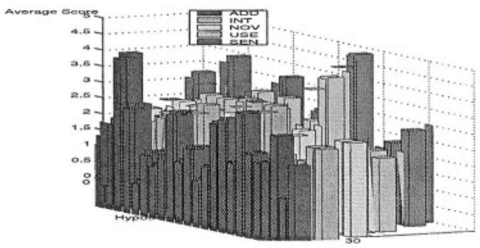


Fig. 2. Human Experts' Assessment of Discovered Patterns

The assessment of individual criteria shows some hypotheses did well with scores above the average on a 1-5 scale. This is the case for hypotheses 11, 16 and 19 in terms of INT, hypotheses 14 and 19 in terms of SEN, hypotheses 1, 5, 11, 17 and 19 in terms of USE, and hypotheses 24 in terms of NOV, etc.

Note also that the assessment seems to be consistent for individual hypotheses across the criteria: hypothesis 19 is well above the average for almost all the criteria (except for NOV), hypothesis 18 always received a score below 2 (25%) except for ADD in which this is slightly higher. Similar situations can be observed for hypotheses 2, 21, etc.

From this automatic evaluation for the discovered patterns, we measured the correlation between the scores of the human subjects and the system's model evaluation. Since both the expert and the system's model evaluated the results considering several criteria, we first performed a normalization aimed at producing a single "quality" value for each hypothesis.

We then calculated the pair of values for every hypothesis and obtained a (Spearman) correlation $r = 0.43$ $(t-test = 23.75, df = 24, p < 0.001)$. From this result, we see that the correlation shows a good level of prediction compared to humans. This indicates that for such a complex task (knowledge discovery), the model's behavior is not too different from the experts'. Note that unlike other approaches, our model was able to do it better without any external linguistic resources.

In order to show what the final hypotheses look like and how the good characteristics and less desirable features as above are exhibited, we picked one of the best hypotheses as assessed by the experts considering the average value of the 5 scores assigned by the user. For example, hypothesis 88 of run 3 looks like:

```
IF goal(show(11511))and method(use(25511))THEN effect(1931,1932)
```

Where the numerical values represent internal identifiers for the arguments and their semantic vectors, and its resulting criteria vector is [0.29,0.18,0.41,0.030,0.28,0.99,0.30,0.50] (the vector's elements represent the values for the criteria relevance, structure, coherence, cohesion, interestingness, plausibility, coverage, and simplicity) and obtained an average expert's assessment of 3.20. In natural-language text, this can roughly be interpreted as:

```
IF  the goal is to show that  the forest restoration ..
AND the method is based on the use of  micro-environments for
    capturing  farm mice..
THEN digestibity "in vitro" should have   an effect on the bigalta
    cuttings..
```

This hypothesis looks more complete than others (goal, methods, etc) but is less relevant than the previous hypothesis despite its close coherence. Note also that the plausibility is much higher than for hypothesis 65, but the other criteria seemed to be a key factor for the human experts.

5 Conclusions

In this paper, a unique approach for evaluation is introduced which deals with semantic and Data Mining issues in a high-level way. In this context, the proposed representation for hypotheses suggests that performing shallow analysis of the documents and then capturing key rhetorical information may be a good level of processing which constitutes a trade off between completely deep and keyword-based analysis of text documents. In addition, the results suggest that the performance of the model in terms of the correlation with human judgments are slightly better than approaches using external resources as in [1]. In particular criteria, the model shows a very good correlation between the system evaluation and the expert assessment of the hypotheses.

In addition, unlike traditional approaches to Text Mining, in this paper we contribute an innovative way of combining additional linguistic information and evolutionary learning techniques in order to produce novel hypotheses which involve explanatory and effective novel knowledge.

The model deals with the hypothesis production and evaluation in a very promising way which is shown in the overall results obtained from the experts evaluation and the individual scores for each hypothesis. However, it is important to note that unlike human experts who have a lot of experience, preconceived concept models and complex knowledge in their areas, the system has done relatively well only exploring the corpus of technical documents and the implicit connections contained in it.

References

1. S. Basu, R. Mooney, K. Pasupuleti, and J. Ghosh. Using Lexical Knowledge to Evaluate the Novelty of Rules Mined from Text. *Proceedings of NAACL 2001 Workshop on WordNet and Other Lexical Resources: Applications, Extensions and Customizations, Pittsburg,* June 2001.
2. Kalyanmoy Deb. *Multi-objective Optimization Using Evolutionary Algorithms.* Wiley, 2001.
3. T. Van Dijk and W. Kintsch. *Strategies of Discourse Comprehension.* Academic Press, 1983.
4. P. Foltz, W. Kintsch, and T. Landauer. The Measurement of Textual Coherence with Latent Semantic Analysis. *Discourse processes,* 25(2):259–284, 1998.
5. A. Freitas. Evolutionary computation. In *Handbook of Data Mining and Knowledge Discovery.* Oxford University Press, 2001.
6. M. Hearst. Automated Discovery of WordNet Relations. In *WordNet: An Electronic Lexical Database,* pages 131–151. MIT Press, 1998.
7. W. Kintsch. Predication. *Cognitive Science,* 25(2): 173–202, 2001.
8. Y. Kodratoff. Applying Data Mining Techniques in Text Analysis. Technical Report unpublished, Laboratoire de Recherche en Informatique (LRI), Inference and Learning Group, Universite Paris Sud, France, 2000.
9. C. Manning and H. Schutze. *Foundations of Statistical Natural Language Processing.* MIT Press, 1999.

10. D. Swanson. On the Fragmentation of Knowledge, the Connection Explosion, and Assembling Other People's ideas. *Annual Meeting of the American Society for Information Science and Technology,* 27(3), February 2001.
11. E. Zitzler and L. Thiele. An Evolutionary Algorithm for Multiobjective Optimisation: The Strength Pareto Approach. Technical Report 43, Swiss Federal Institute of Technology (ETH), Switzerland, 1998.

An Electronic Assistant for Poetry Writing

Nuno Mamede*, Isabel Trancoso*, Paulo Araújo[†], and Céu Viana[♮]

* INESC-ID Lisboa/IST
[†]INESC-ID Lisboa/ISEL
[♮] CLUL
L^2F - Spoken Language Systems Lab, INESC ID Lisboa,
Rua Alves Redol, 9,1000-029 Lisboa, Portugal
{Nuno.Mamede, Isabel.Trancoso, Paulo.Araujo,
Ceu.Viana}@l2f.inesc-id.pt
http://www.l2f.inesc.pt

Abstract. The ultimate goal of the poetry assistant currently under development in our lab is an application to be used either as a poetry game or as a teaching tool for both poetry and grammar, including the complex relationships between sound and meaning. Until now we focused on the automatic classification of poems and the suggestion of the ending word for a verse. The classification module is based on poetic concepts that take into account structure and metrics. The prediction module uses several criteria to select the ending word: the structural constraints of the poem, the grammatical category of the words, and the statistical language models obtained from a text corpus.

The first version of the system, rather than being self-contained, is still based on the use of different heterogeneous modules. We are currently working on a second version based on a modular architecture that facilitates the reuse of the linguistic processing modules already developed within the lab.

1 Introduction

This paper describes the early stages of the development of a system to assist poetry writing. Its main goals are the classification of poems and the suggestion of verses' ending words. The poem classification is based on poetic concepts, such as the number of verses in the stanzas; the stanzaic form; the number of syllables on the verses; and the rhyme scheme. The suggestion of the ending word of a verse may be based on different selection criteria or combination of criteria: the structural constraints of the poem, the grammatical category of the words, and the statistical language models obtained from a text corpus.

With such a system, we intend on one hand to help understanding the structure and rhyme of poems, which may be important to improve the reading aloud capabilities of students and, on the other hand, to encourage them to write their own poems.

Although there are some Internet sites that publish and discuss poetry for Portuguese [1,2], there are no computer programs to assist the process of writing poems for our language and we could not find any framework that handles rhyme correctly. A rhyme dictionary [3] exist, but it only takes into account the final letters of the words, not their pronunciation.

For the English language, we could find some interesting word games where the user can make poems with those words [4], programs that allow the generation of nonsense poems based on syntax definitions [5], and others that generate poetry based on models created from poetry corpora [6].

One of the interesting features of our application is the fact that it relies on tools which, with few exceptions, have already been implemented in the lab. Hence, the first version of the system, rather than being self-contained, is still based on the use of different heterogeneous modules. We are currently working on a second version based on a modular architecture that facilitates the reuse of the already existent linguistic processing modules.

We shall start the description of our poetry assistant system with the classification module (Section 2), focusing on the submodule that presents some important research challenges - the metric syllable counter. Section 3 describes the prediction module, and the selection criteria that can be combined to suggest the missing words. Finally, Section 4 describes the new system architecture we are currently working on.

The two following sections report informal evaluation experiments conducted using a very small test corpus that was manually classified. In addition to around 20 stanzas from two very well known Portuguese poets, the corpus also includes around 200 poems written by children (ages 7-9). The corpus collection task is by no means complete yet. In fact, it has recently been enlarged with the poems collected in the scope of a national project dealing with digital spoken books [7]. The informal tests described below, however, do not yet include this recent addition.

2 Classification Module

This module currently takes as input a finished poem, and yields as output the number of lines and stanzas, the stanzaic form, and the rhyme scheme.

One of the main tools used by this module is a Grapheme-to-Phone conversion tool. Several approaches have been developed in the Lab for this purpose: based on rules, neural networks [8], and classification and regression trees [9]. The current GtoP module of the DIXI+ system is based on CARTs trained on a lexicon and has a word error rate of 3.8%, slightly higher than the rules system. Our latest efforts in terms of GtoP conversion have expressed the original rules as FSTs [10].

The GtoP tool yields the phonetic transcription of each verse, taking into account generic word co-articulation rules. It outputs a string of symbols of the SAMPA phonetic alphabet for European Portuguese [11], as well as lexical stress markers.

The following is an example of the output of the classification module, using the first stanza of the most famous epic poem in Portuguese (Os Lus'adas, by Camões, XVI century):

As armas e os barões assinalados
Que da ocidental praia lusitana
Por mares nunca dantes navegados
Passaram ainda além da Taprobana,
Em perigos e guerras esforçados
Mais do que prometia a força humana,

E entre gente remota edificaram
Novo Reino, que tanto sublimaram;

```
Summary:
  lines: 8
  verses: 8
  stanzas: 1
  rhyme:  [A,B,A,B,A,B,C,C]
  syllables: [10,10,10,10,10,10,10,10]
  Classification:  “Ottava  rima”
```

Although the relevance of the phonetic transcription tool for the rhyme classification module is not well illustrated by this example (where the same conclusions over rhyme could be derived from the orthography alone), this is not always the case. The above illustrated rhyming criterion requires perfect rhymes, where the last stressed vowel and succeeding sounds are identical. An alternative criterion could be rhyming only in terms of the two last syllabic nuclei (e.g. *dentro* and *vento, silêncio* and *cinzento*).

2.1 Metric Structure

One of the main research challenges of this project is dealing with the metric structure of the poems. In fact, counting metric syllables (up to the last stressed syllable) is not an easy task, if one takes into account word co-articulation phenomena and different possibilities of phonological reduction.

In the current version of the classification module, syllabic boundary markers are placed a posteriori on the SAMPA string using a simple script based on generic rules. As an example of its application to the sixth verse in the above poem, we obtain the correct number of 10 metric syllables (syllable boundaries are marked by ”$” or by ”, for stressed syllables), accounting for vowel coalescence and diphthonguisation (7th and 9th syllables, respectively).

```
“majZ$du$k@$pru$m@”ti$a“for$s6w”m6$n6
```

We are currently working on another version which allows for some phonological variation not accounted for by the GtoP rules, yielding for each verse a range of values, rather than a single value, for the number of metric syllables. This work is closely related with our past work on pronunciation variation for phone alignment [12] where we have modeled variants that depend on the local immediate segmental context through rules implemented via finite state transducers. The main phonological aspects that the rules are intended to cover are: vowel devoicing, deletion and coalescence, voicing assimilation, and simplification of consonantal clusters, both within words and across word boundaries. Some common contractions are also accounted for, with both partial or full syllable truncation and vowel coalescence. Vowel reduction, including quality change, devoicing and deletion, is specially important for European Portuguese. In fact, as a result of vowel deletion, rather complex consonant clusters can be formed across word boundaries.

Notice that vowel deletion can be deliberately marked in the orthography by the authors themselves (e.g. *já como um ‘stigma* by David Mourão Ferreira in “Dos anos

Trinta", in which the first vowel "e", pronounced as a schwa and often deleted, is not included in the orthography).

2.2 Meter

The identification of the stressed syllables is also important for classifying the meter - rising (iambs and anapests), or falling (trochees and dactyls), depending on having one or two unstressed syllables followed by a stressed one or a stressed syllable followed by one or two unstressed syllables. It is also important for classifying each verse according to the position of the stressed syllable of the last rhyming word (oxytone words have the accent in the last syllable; paroxytone ones have the accent in the penultimate syllable; and proparoxytone words have the accent in the antepenult syllable). Proparoxytone verses are much less frequent.

The meter classification part is not yet implemented.

2.3 Informal Evaluation

An informal evaluation of the classification module was done using our still small test corpus. The results obtained with the already implemented modules are very positive.

A byproduct of this work was the development of the first electronic Portuguese rhyme dictionary, in which the search for rhyming words is made based on the word's phonetic transcription.

3 Word Prediction Module

This module takes as input an incomplete poem, for which the user may request the suggestion of a word that rhymes with the other verses of the stanza. The suggestion may be based on different selection criteria or combination of criteria, previously defined by the user. The selection criteria should include:

- words rhyming with the last verse
- words rhyming with a given verse
- metrical atructure of the verse
- words belonging to a given grammatical category (noun, adjective, verb,...)
- words belonging to a given ontology (animal, tree, flower, job, country,...)
- words that may follow the last word written by the user, according to some statistical language model

The list of words that satisfy all the criteria defined by the user may be too large to allow its presentation to the user. Hence the module selects the 10 best words in this list.

The prediction module uses the same GtoP and metric syllable counting tools as the classification module. In addition, it uses some linguistic analysis modules already available in the lab, for the generation of possible word classes that may follow the last word of the incomplete poem:

- Smorph: Morphological Analyser [13]
- PAsMo: Post-Morphological Analysis [14]
- SuSAna: Surface Syntactic Analyser [15]

The submodule related to the ontology criterion is not yet implemented. The submodule that involves statistical language models uses n-gram models built using the CMU-Cambridge Statistical Language Modeling toolkit [16] which provides support for n-grams of arbitrary size, and support for several discounting schemes. Rather than retraining new models with our restricted poetry corpus, we used the existent n-grams currently built for a broadcast news task (57k unigrams, 6M bigrams, 11M trigrams,...).

As an example of the application of the word prediction module, consider the following poem by Antonio Aleixo:

A quem prende a água que corre
é por si próprio enganado;
O ribeirinho não morre,
vai correr por outro lado.

Let us suppose that the last word *(lado)* was not included. By using a bigram model criterion, the first 10 words suggested would be (by decreasing order of frequency):

lado, dos, a, de, que, o, para, e, dia, em, ...

On the other hand, by using a rhyme criterion, the first 10 suggested words would be:

lado, passado, resultado, dado, deputado, avanado,
machado, demasiado, obrigado, advogado, ...

The application of the number of metric syllables criterion (7 in this case, for the other verses), together with the above criteria, restricts the list of 2-syllable suggested words to:

lado, dado, fado

The missing word was thus successfully suggested by the prediction module.

3.1 Informal Evaluation

An informal evaluation of this module was conducted using our still very small poetry corpus as test corpus and repeating the above procedure of removing the last word of each stanza. Our preliminary results led us to conclude that n-grams of higher order (3-grams, 4-grams,...) are not very effective, as the missing word is not always among the first 10 most frequent n-grams, and in some cases it is not present at all.

The effectiveness of the use of structural constraints was also compared with the use of the bigram models. Our first tests intended to compare the bigram criterion with the one using the number of metric syllables. Our preliminary results led us to conclude that in the 10 most frequent words produced by the second criterion, there are more words that could be used as a substitute of the missing word than in the corresponding list produced by the bigram criterion. Similar preliminary conclusions could be drawn when comparing the use of the bigram criterion with the rhyme one: the latter yielded a larger number of words that could be used as a replacement of the missing word. The last bunch of tests used a combination of the two structural constraints: number of syllables and rhyme. In this case, most of the words in the 10-best list could be used as a replacement.

4 System Architecture

As stated above, one of the interesting features of our poetry assistant is the reuse of several already available linguistic processing modules. Hence it was particularly important to choose a suitable architecture. We wanted it to be domain independent, language independent, distributed (over the internet) and normalized over each type of functional module. Moreover, we would like to have the possibility of aggregating existing modules to obtain a "new" module with a "new" interface.

These design goals led to the development of GalInHa [17], a web-based user interface for building modular applications that enables users to access and compose modules using a web browser. GalInHa is the short name for Galaxy Interface Handler, inspired as the name indicates on GALAXI II [19], the architecture defined at MIT and used by the DARPA Communicator initiative. GALAXI II is a distributed system with a client-server architecture. It concentrates the communication between modules in a single module, the hub, using a standardized protocol.

Since GalInHa deals with Galaxy process chains, all that is needed for the development of the poetry assistant is to define the corresponding chain and to connect the modules. Some of the needed linguistic analysis modules are already available through GalInHa (E.g. Smorph, PAsMo, SuSAna), although they accept/produce different data formats, so additional modules were needed to provide data format conversion.

All that is needed to connect the new modules is to include an existing XSLT [20] processor into GalInHa. For that, we have to write a wrapper to call external applications. The other modules are currently being adapted to the GalInHa architecture.

`Galinha`'s Galaxy-based general architecture is shown on figure 1.

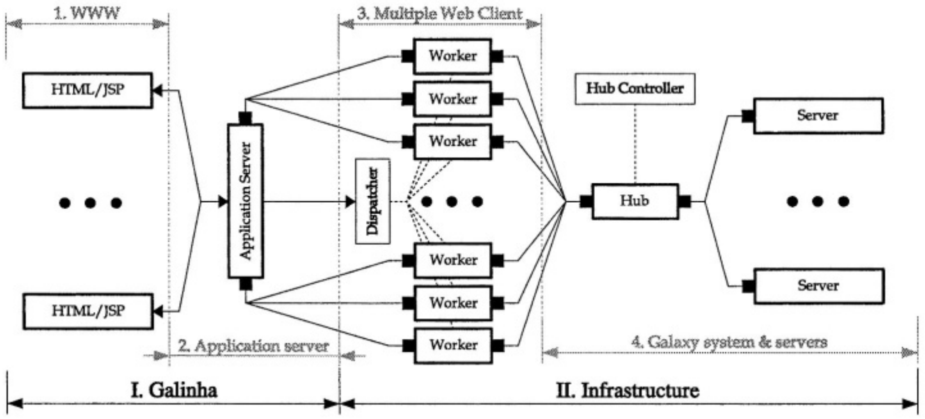


Fig. 1. The Galaxy infrastructure, control servers, and user-side levels

The application server is one of the interface's key components. It provides the runtime environment for the execution of the various processes within the web application. Moreover, it maintains relevant data for each user, guarantees security levels and manages access control. The interface also uses the application server as a bridge between the interface's presentation layer (HTML [24]/JavaScript [21], at the browser level) and the infrastructure.

The presentation layer consists of a set of PHP scripts and classes [22] and related pages. It is built from information about the location (host and port) of the MultipleWebClient that provides the bridge with the Galaxy system the user wants to contact; and from XML [23] descriptions of the underlying Galaxy system (provided by the hub controller). This is a remake of a previous Java/servlets-based interface.

Besides allowing execution of services on user-specified data, the interface allows users to create, store, and load service chains. Service chains are user-side service- or program sequences provided by the servers connected to the infrastructure: each service is invoked according to the user-specified sequence. Service chains provide a simple way for users to test sequences of module interactions without having to actually freeze those sequences or build an application. The interface allows not only inspection of the end results of a service chain, but also of its intermediate results. Service chains may be stored locally, as XML documents, and may be loaded at any time by the user. Even though, from the infrastructure's point of view, service chains simply do not exist, selected service chains may be frozen into system-side programs and become available for general use. Other developments on the user side are currently being studied to address sharing of chain configurations without infrastructure support.

Modules may be included in the infrastructure in two ways: the first is to create the module anew or to adapt it so that it can be incorporated into the system; the second is to create a capsule for the existing module – this capsule then behaves as a normal Galaxy server would.

Whenever possible or practical, we chose the second path. Favoring the second option proved a wise choice, since almost no changes to existing programs were required. In truth, a few changes, mainly regarding input/output methods, had to be made, but these are much simpler than rebuilding a module from scratch: these changes were caused by the requirement that some of the modules accept/produce XML data in order to simplify the task of writing translations. This is not a negative aspect, since the use of XML as intermediate data representation language also acts as a normalization measure: it actually makes it easier for future users to understand modules' inputs and outputs. It also eases the eventual conversion between module's outputs and inputs (also needed on a command-line approach).

5 Conclusions and Future Trends

This paper reported on-going work on the development of a poetry assistant system which, besides involving many already existing tools in the lab, also presents some interesting research challenges not only in terms of the overall architecture, but namely in terms of metric structure. The nest stage of the project will focus on rhythm and intonation and on how these may convey differences of meaning.

The last stage of the project will be devoted to evaluation. We plan to enlarge our test corpus in order to conduct some formal evaluation of the two modules (also assessing response time). In addition, we plan to do some user evaluation, using a panel of primary and high-school teachers and students, in order to evaluate the performance and the usability of the interface.

We hope that this system would be a valuable e-learning tool which may be used in classrooms or at home, to classify poems, help understand the concepts of structure and rhyme, and encourage students to write their own poems by suggesting the next words.

References

1. *Geração Poesia,* World Wide Web Consortium (W3C), 2001, see: `http://www.geracaopoesia.meublog.com.br.`
2. *Projecto Vercial,* 1996-2004, see: `http://www.ipn.pt/opsis/litera.`
3. *Dicionário de Rimas Poéticas,* Pretor Informática e Sistemas Ltda, 2000, `http://www.lemon.com.br.`
4. *Magnetic Poetry Kit,* Magnetic Poetry, Inc., 2000, see: `http://www.magneticpoetry.com.`
5. A. Chachanashvili, *Dada Poem Generator,* 2000.
6. R. Kurzweil, *Ray Kurzweil's Cybernetic Poet,* CyberArt Technologies, 1999, see: `http://www.kurzweilcyberart.com/poetry/rkcp-overview.php3.`
7. A. Serralheiro, I. Trancoso, D. Caseiro, T. Chambel, L. Carriço, and N. Guimarães, "Towards a repository of digital talking books," in *Proc. Eurospeech '2003,* Geneva, Switzerland, Sept. 2003.
8. I. Trancoso, M. Viana, F. Silva, G. Marques, and L. Oliveira, "Rule-based vs. neural network based approaches to letter-to-phone conversion for portuguese common and proper names," in *Proc. ICSLP '94,* Yokohama, Japan, Sept. 1994.
9. L. Oliveira, M. C. Viana, A. I. Mata, and I. Trancoso, "Progress report of project dixi+: A portuguese text-to-speech synthesizer for alternative and augmentative communication," FCT, Tech. Rep., Jan. 2001.
10. D. Caseiro, I. Trancoso, L. Oliveira, and C. Viana, "Grapheme-to-phone using finite state transducers," in *Proc. 2002 IEEE Workshop on Speech Synthesis,* Santa Monica, CA, USA, Sept. 2002.
11. *SAMPA (SAM Phonetic Alphabet),* Spoken Language Systems Lab (L2F), see: `http://www.l2f.inesc-id.pt/resources/sampa/sampa.html.`
12. I. Trancoso, D. Caseiro, C. Viana, F. Silva, and I. Mascarenhas, "Pronunciation modeling using finite state transducers," in *Proc. 15th International Congress of Phonetic Sciences (ICPhS'2003),* Barcelona, Spain, Aug. 2003.
13. S. Ait-Mokhtar, "L'analyse présyntaxique en une seule étape," Ph.D. dissertation, Universit Blaise Pascal, GRIL, Clermont-Ferrand, 1998.
14. J. L. Paulo, "PAsMo – Pós-AnáliSe MOrfológica," Laboratório de Sistemas de Língua Falada **(L^{2}F – INESC-ID),** Lisboa, Relatório Técnico, 2001.
15. F. M. M. Batista, "Análise sintáctica de superfície," Master's thesis, Instituto Superior Técnico, Universidade Técnica de Lisboa, July 2003.
16. P. Clarkson and R. Rosenfeld, "Statistical language modeling using the cmu-cambridge toolkit," in *Proc. Eurospeech '97,* Greece, Sept. 1997.
17. D. M. de Matos, J. L. Paulo, and N. J. Mamede, "Managing Linguistic Resources and Tools," in Lecture Notes in Artificial Inteligence, no. 2721, Springer-Verlag, 2003, pp. 135–142.
18. Massachusetts Institute of Technology (MIT), The MITRE Corporation. *Galaxy Communicator (DARPA Communicator).* See: `http://communicator.sf.net.`
19. S. Seneff, E. Hurley, R. Lau, C. Pao, P. Schmid, and V. Zue, "Galaxy-ii: A reference architecture for conversational system development," in *Proc. ICSLP '98,* Sydney, Australia, Dec. 1998.

20. W3C, *The Extensible Stylesheet Language,* World Wide Web Consortium (W3C), 2001, see: www.w3.org/Style/XSL.
21. ECMA International, Geneva, Switzerland. *Standard ECMA-262 – ECMAScript Language Specification,* 3rd edition, December 1999. See also: www.ecma.ch.
22. PHP Group. *PHP Hypertext Processor.* See: www.php.net.
23. World Wide Web Consortium (W3C). *Extensible Markup Language (XML).* See: www.w3.org/XML.
24. World Wide Web Consortium (W3C). *HyperText Markup Language (HTML).* See: www.w3.org/MarkUp.
25. David M. de Matos, Alexandre Mateus, João Graça, and Nuno J. Mamede. Empowering the user: a data-oriented application-building framework. In *Adj. Proc. of the 7th ERCIM Workshop "User Interfaces for All",* pages 37–44, Chantilly, France, October 2002. European Research Consortium for Informatics and Mathematics.

Improving the Performance of a Named Entity Extractor by Applying a Stacking Scheme

José A. Troyano, Víctor J. Díaz, Fernando Enríquez, and Luisa Romero

Department of Languages and Computer Systems,
University of Seville,
Av. Reina Mercedes s/n 41012, Sevilla (Spain)
troyano@lsi.us.es

Abstract. In this paper we investigate the way of improving the performance of a Named Entity Extraction (NEE) system by applying machine learning techniques and corpus transformation. The main resources used in our experiments are the publicly available tagger TnT and a corpus of Spanish texts in which named entities occurrences are tagged with BIO tags. We split the NEE task into two subtasks 1) Named Entity Recognition (NER) that involves the identification of the group of words that make up the name of an entity and 2) Named Entity Classification (NEC) that determines the category of a named entity. We have focused our work on the improvement of the NER task, generating four different taggers with the same training corpus and combining them using a stacking scheme. We improve the baseline of the NER task ($F_{\beta=1}$ value of 81.84) up to a value of 88.37. When a NEC module is added to the NER system the performance of the whole NEE task is also improved. A value of 70.47 is achieved from a baseline of 66.07.

1 Introduction

Named Entity Extraction involves the identification of words that make up the name of an entity, and the classification of this name into a set of categories. For example, in the following text, the words "Juan Antonio Samaranch" are the name of a person, the word "COI" is an organization name, "Río de Janeiro" is the name of a place and, finally, "Juegos Olímpicos" is an event name:

El presidente del *COI, Juan Antonio Samaranch,* se sumó hoy a las alabanzas vertidas por otros dirigentes deportivos en *Río de Janeiro* sobre la capacidad de esta ciudad para acoger unos *Juegos Olímpicos.*

In order to implement a system that extracts name entities from plain text we have to meet with two different problems, the recognition of a named entity and its classification. Named Entity Recognition (NER) is the identification of the word sequence that forms the name of an entity, and Named Entity Classification (NEC) is the subtask in charge of deciding which is the category assigned to a previously recognized entity.

There are systems that perform both subtasks at once. Other systems, however, make use of two independent subsystems to carry out each subtask sequentially. The second architecture allows us to choose the most suitable technique to each subtask. Named entity recognition is a typical grouping task (or chunking) while choosing its category is a classification problem. In practice, it has been shown [3] that the division into two separate subtasks is a very good option.

Our approach to the NEE problem is based on the separate architecture. In the development of the NER module we have followed the next steps:

- To eliminate from the corpus the information relating to the categories, leaving only the information about the identification of named entity boundaries.
- To apply three transformations to the recognition corpus. Thus we have different views of the same information which enable the tagger to learn in different ways.
- To train TnT with the four corpora available for the NER task (the original and the results of the three transformations).
- To combine the results of the four taggers in order to obtain a consensual opinion. This combination has been carried out applying a stacking scheme, where the results of the different models are used to generate a training database employed in a second stage of learning.

The NEC module has been implemented by a classifier induced from a training database. The database is obtained calculating a feature vector for each entity in the training corpus. The NEC classifier and the classifier used in the stacking scheme have been built with algorithms of the weka package [14].

Experiments show that the three transformations improve the results of the NER task, and that system combination achieves better results than the best of the participant models in isolation. This improvement in the NER task repels positively in the performance of the NEE task. When a NEC module is applied to the previously recognized entities, an improvement of more than four points is achieved with respect to the baseline defined for the NEE task.

The organization of the rest of the paper is as follows. The second section presents the resources, measures and baselines used in our experiments. In section three we show how to improve the NER subtask applying corpus transformations. In section four we describe how to combine the results of the four NER systems with a stacking scheme. Section five presents the NEC module and the results of its use in conjunction with the best NER system. Finally, in section six we draw the final conclusions and point out some future work.

2 Resources and Baselines

In this section we describe the main resources used in our experiments, and the baselines we have employed to measure the improvements achieved.

2.1 The Corpus and the Tagger

This corpus provides a wide set of named entity examples in Spanish. It was used in the NER task of CoNLL-02 [12]. The files are:

- Training corpus with 264715 tokens and 18794 entities
- Test corpus with 52923 tokens and 4315 entities

BIO notation is used to denote the limits of a named entity. The initial word of a named entity is tagged with a B tag, and the rest of words of a named entity are tagged with I tags. Words outside an entity are denoted with an O tag. There are four categories in the corpus taxonomy: PER (people), LOC (places), ORG (organizations) and MISC (rest of entities), so the complete set of tags is {B-LOC, I-LOC, B-PER, I-PER, B-ORG, I-ORG, B-MISC, I-MISC, O}.

The NER task does not need the category information, so we have simplified the tag set removing the category information from the tags. Figure 1 shows a fragment of the original corpus, and its simplified version used in the NER task.

Word	Tag
El	O
presidente	O
del	O
COI	B-ORG
,	O
Juan	B-PER
Antonio	I-PER
Samaranch	I-PER
,	O
se	O
sumó	O
...	...

Word	Tag
El	O
presidente	O
del	O
COI	B
,	O
Juan	B
Antonio	I
Samaranch	I
,	O
se	O
sumó	O
...	...

Fig. 1. Original Corpus and Corpus Tagged Only for the Recognition Subtask

We have choosen the tagger TnT as basis for developing the NER systems presented in this paper. TnT [1] is one of the most widely used re-trainable tagger in NLP tasks. It is based upon second order Markov Models, consisting of word emission probabilities and tag transition probabilities computed from trigrams of tags.

2.2 Measures and Baselines

The measures used in our experiments are, *precision, recall* and the overall measure $F_{\beta=1}$. These measures were originally used for Information Retrieval evaluation purposes, but they have been adapted to many NLP tasks.

Precision is computed according to the number of correctly recognized entities, and *recall* is defined as the proportion of the actual entities that the system has been able to recognize:

$$Precision = \frac{\text{correctly extracted entities}}{\text{extracted entities}}$$

$$Recall = \frac{\text{correctly extracted entities}}{\text{actual entities}}$$

Finally, $F_{\beta=1}$ combines recall and precision in a single measure, giving to both the same relevance:

$$F_{\beta=1} = \frac{2\ Precision\ Recall}{Precision + Recall}$$

We will trust in $F_{\beta=1}$ measure for analyzing the results of our experiments. It is a good performance indicator of a system and it is usually used as comparison criterion. Table 1 shows the results obtained when TnT is trained with the original corpus*(NEE_baseline)*and with its simplified version used in the NER subtask *(NER_baseline),* we will adopt these results as the baselines for further experiments in this paper.

Table 1. Baselines. NEE and NER Results with TnT

	Precision	Recall	$F_{\beta=1}$
NEE_baseline	66.28%	65.85%	66.07
NER_baseline	81.40%	82.28%	81.84

The NER baseline is much higher than the NEE baseline because the NER problem is simpler than the whole NEE task. In this paper we will take the approach of improving the NER subtask to build an NEE system (adding a NEC module) that improves the NEE baseline.

3 Improving NER Task Through Corpus Transformation

It seems logical to think that if we have more information before taking a decision we have more possibilities of choosing the best option. For this reason we have increased the number of models as a way of improving the performance of the NER task. There are two obvious ways of building new models: using new training corpora or training other taggers with the same corpus. We have tried a different approach, defining three transformations that give us three additional versions of the training corpus. Transformations can be defined to simplify the original corpus or to add new information to it. If we simplify the corpus, we reduce the number of possible examples and the sparse data problem will be smoothed. On the other hand if we enrich the corpus, the model can use new information to identify new examples not recognized by the original model.

3.1 Vocabulary Reduction

This transformation discards most of the information given by words in the corpus, emphasizing the most useful features for the recognition. We employ a technique similar to that used in [10] replacing the words in the corpus with tokens that contain relevant information for recognition. The goals pursued are:

- To stand out certain typographic features of words, like capitalization, that can help in deciding if a word is part of a named entity or not.
- To give more relevance to words that can help in the identification of a named entity.
- To group several words of the vocabulary into a single entry.

Apart from typographic information there are other features that can be useful in the identification of entities, for example non-capitalized words that frequently appear before, after or inside named entities. We call them trigger words and they are of great help in the identification of entity boundaries.

Both pieces of information, trigger words and typographic clues, are extracted from the original corpus through the application of the following rules:

- Each word is replaced by a representative token, for example, *_starts_cap_* for words that start with capital letters, *_lower_* for words that are written in lower case letter, *_all_cap_* if the whole word is upper case, etc. These word patterns are identified using a small set of regular expressions.
- Not all words are replaced with its corresponding token, trigger words remain as they appear in the original corpus. The list of trigger words is computed automatically counting the words that most frequently appear around or inside an entity.

Figure 2 shows the result of applying these rules to the corpus fragment of Figure 1. Vocabulary reduction leads to an improvement in the performance of the NER subtask. The results of the experiment *TnT-V* are presented in Table 2, we can see that TnT improves from 81.84 to 83.63.

3.2 Change of Tag Set

This transformation does not affect to words but to tags. The basic idea is to replace the original BIO notation with a more expressive one that includes information about the words that usually end a named entity. The new tag set has five tags, the three original (although two of them change slightly their semantic) plus two new tags:

- B, that denotes the beginning of a named entity with more than one word.
- BE, that is assigned to a single-word named entity.
- I, that is assigned to words that are inside of a multiple-word name entity, except to the last word.
- E, assigned to the last word of a multiple-word named entity.
- O, that preserves its original meaning: words outside a named entity.

The new tag set gives more relevance to the position of a word, forcing the tagger to learn which words appear more frequently at the beginning, at the end or inside a named entity. Figure 2 shows the result of applying this new tag set to the corpus fragment of Figure 1. Changing the tag set also leads to better results in the NER task than those obtained with the original corpus. The results of the experiment *TnT-N* are showed in Table 2. In this case, TnT improves from 81.84 to 84.59, the best result of the three transformations studied.

3.3 Addition of Part-of-Speech Information

Unlike previous corpus transformations, in this case we will make use of external knowledge to add new information to the original corpus. Each word will be replaced with a compound tag that integrates two pieces of information:

- The result of Applying the First Transformation (Vocabulary Reduction).
- The part of speech (POS) tag of the word.

In order to obtain the POS tag of a word we have trained TnT with the tagged corpus CLiC-TALP [4]. This corpus is a one hundred thousand word collection of samples of written language, it includes extracts from newspapers, journals, academic books and novels. Figure 2 shows the result of the application of this transformation to the corpus fragment of Figure 1. Adding part of speech information also implies an improvement in the performance of TnT in the NER task. Table 2 presents the results of the experiment *TnT-P,* in this case TnT reaches an $F_{\beta=1}$ measure of 83.12.

Word	Tag
El	O
presidente	O
del	O
_all_cap_	B
,	O
_starts_cap_	B
_starts_cap_	I
_starts_cap_	I
,	O
se	O
lower	O
...	...

a) Vocabulary reduction.

Word	Tag
El	O
presidente	O
del	O
COI	BE
,	O
Juan	B
Antonio	I
Samaranch	E
,	O
se	O
sumó	O
...	...

b) Change of tag set.

Word	Tag
El_det_	O
presidente_noun_	O
del_prep_	O
_all_cap__noun_	B
,_punt_	O
_starts_cap__noun_	B
_starts_cap__noun_	I
_starts_cap__noun_	I
,_punt_	O
se_pron_	O
_lower__verb_	O
...	...

c) Addition of POS information.

Fig. 2. Result of Applying Transformations to the Corpus Fragment Showed in Figure 1

4 Improving the NER Task Through System Combination

The three transformations studied cause an improvement in the performance of the NER task. But we still have room for improvement if instead of applying the transformations separately we make them work together. A superficial analysis of the texts tagged by each model shows that not all the models make the same mistakes. There are some "very difficult" examples that are not recognized by any model, but many of the mistakes can be corrected taking into account the tag proposed by other models. System combination is not a new approach in

NLP tasks, it has been used in several problems like part of speech tagging [6], word sense disambiguation [8], parsing [7] and noun phrase identification [11].

4.1 Stacking

Stacking consists in applying machine learning techniques for combining the results of different models. The main idea is to build a combining system that learns the way in which each model is right or makes a mistake. In this way, the final decision is taken according to a pattern of correct and wrong answers.

We need a training database to be able of learning the way in which every model is right or wrong. Each register in the training database includes all the tags proposed by the participant models for a given word and the actual tag. In order to enrich the training database we have included in the registers the tags of the two previous and the two following words, this way we take into account not only the information associated to a word but also information about its context. We make the database independent of the training and test corpus using an additional corpus of 51533 words to generate the registers. Figure 3 presents an extract of the generated database. For each model there are five tags that correspond, respectively, to the two previous words, the word in question and the two following words. The last tag of each register is the actual tag of the word represented by the register.

TnT	TnT-V	TnT-N	TnT-P	Tag
OOOOB	OOOOB	OOOOB	OOOOB	O
OOOBO	OOOBI	OOOBO	OOOBO	O
OOBOB	OOBII	OOBOB	OOBOB	B
OBOBO	OBIIO	OBOBO	OBOBO	O
BOBOO	BIIOO	BOBOO	BOBOO	B
OBOOO	IIOOO	OBOOO	OBOOO	O

Fig. 3. Extract of the Generated Database

Apart from allowing the use of heterogeneous information, machine learning has another important advantage over voting: it is possible to choose among a great variety of schemes and techniques to find the most suitable one to each problem. *Bagging* [2] is one of these schemes, it provides a good way of handling the possible bias of the model towards some of the examples of the training database. Bagging is based on the generation of several training data sets taking as base a unique data set. Each new version is obtained by sampling with replacement the original database. Each new data set can be used to train a model and the answers of all the models can be combined to obtain a joint answer. The joint answer is usually obtained by voting. In the experiment *TnT-Stack* we have applied a bagging scheme (using decision trees [9] as base classifier) to combine the results given by *TnT-V, TnT-N* and *TnT-P*. Table 2 shows the results of

Table 2. Results of NER Experiments

	Precision	Recall	$F_{\beta=1}$
NER_baseline	81.40%	82.28%	81.84
TnT-V	81.76%	85.59%	83.63
TnT-N	85.04%	84.15%	84.59
TnT-P	81.51%	84.79%	83.12
TnT-Stack	88.68%	88.05%	88.37

this experiment. With this system we obtain the best result (88.37), with an improvement of 6.53 points with respect to the baseline for the NER subtask. There are other authors that also propose stacking as a way of improve the performance of NEE systems, for example [5] and [15]. In both cases they use several taggers to obtain the different opinions combined through the stacking scheme. In this sense, the main contribution of our work is the use of corpus transformation to obtain the different models needed to apply stacking. This way we have the variability necessary to apply system combination without using several training corpora or several taggers.

5 The NEC Module

After the NER stage, we have a text in which possible entities have been identified without specifying the class they belong. At this point, the named entity extractor is completed with a NEC module implemented by a classifier induced from a training database. The database is obtained by calculating a feature vector for each entity in the training corpus. The features used to generate the vectors are the following:

1. Orthographic: we check if an entity contains words that begin with capital letters, or if they contain digits, Roman numbers, quotes, etc.
2. Length and Position: we use as feature the length in words of an entity, as well as the relative position within the phrase.
3. Suffixes: frequent suffixes for each category are calculated from the training corpus.
4. Contexts: relevant words for each category are calculated in a window of three words around the entities found in the training corpus.
5. Content Words: for each category, the set of significant words is calculated eliminating stop words and those in small letters from the examples of the training corpus.

We have used a Support Vector Classifier [13] to implement this classification task. With this method a binary classifier is defined through an hyperplane that optimally divides the classes. Multi-class classifiers can be built combining binary classifiers with a pairwise ensemble scheme. It has been shown that the optimal hyperplane is determined by only a small fraction of the data points, the so-called

Table 3. Results of NEE Experiments

	Precision	Recall	$F_{\beta=1}$
NEE_baseline	66.28%	65.85%	66.07
NEC_alone	78.22%	78.22%	78.22
TnT-Stack-NEC	70.72%	70.22%	70.47

support vectors. The support vector classifier training algorithm is a procedure to find these vectors.

Table 3 shows the standalone performance of the NEC module (assuming no NER errors) and the results of its application to the best NER system studied in this paper (experiment *TnT-Stack-NEC*). An improvement of more than four points with respect to the baseline defined for the NEE is obtained.

6 Conclusions and Future Work

In this paper we have shown that the performance of a named entity extraction system can be improved by applying a stacking scheme. We have split the NEE task into two subtasks: recognition and classification. Our work has been focused on demonstrating that the performance of the NER task can be improved by combining different NER systems that have been obtained with only one tagger (TnT) and one training corpus. The variability necessary to be able to apply system combination has been achieved applying transformations to the training corpus. The baseline for the NER task is improved from 81.84 to 88.37. This performance is similar to state of the art recognizers, with comparable results to those obtained by one of the best NER systems for Spanish texts [3].

As additional conclusion we have demonstrated that the improvement of the NER task also entails an improvement in the complete extraction task. The experiments demonstrate that the baseline defined for the NEE task is surpassed with clarity, improving from 66.07 to 70.47.

Much future work remains. Recognition results are very good but the classification ones are still poor, we have implemented a very simple NEC module and there are still room for improvement in this aspect including new features and experimenting with new learning methods. We are also interested in applying the ideas of this paper to the extraction of entities in specific domains. In this kind of tasks the knowledge about the domain could be incorporated to the system via new transformations. We also plan to take advantage of system combination to help in the construction of annotated corpus, using the jointly assigned tag as agreement criterion in co-training or active learning schemes.

Acknowledgments

We would like to thank the groups CLiC of the University of Barcelona, and TALP of the Polytechnic University of Cataluña, for letting us use the corpus CLiC-TALP.

References

[1] Brants, T.: TnT. A statistical part-of-speech tagger. In *Proceedings of the 6th Applied NLP Conference (ANLP00).* USA (2000)224–231

[2] Breiman, L.: Bagging predictors. In *Machine Learning Journal* 24(1996) 123–140

[3] Carreras, X., L. Màrquez y L. Padró: Named Entity Extraction using AdaBoost. In *CoNLL02 Computational Natural Language Learning.* Taiwan (2002) 167–170

[4] Civit, M.: Guía para la anotación morfosintáctica del corpus CLiC-TALP. *X-TRACT Working Paper WP-00/06.* (2000)

[5] Florian, R.: Named entity recognition as a house of cards: Classifier stacking. In *CoNLL-2002. Computational Natural Language Learning.* Taiwan (2002) 175–178

[6] Halteren, v. H., Zavrel, J. , Daelemans, W.: Improving accuracy in word class tagging through the combination of machine learning systems. *Computational Linguistics 27* (2001) 199–230

[7] Henderson, J. C., Brill, E.: Exploiting diversity in natural language processing. Combining parsers. In *1999 Joint Sigdat Conference on Empirical Methods in Natural Language Processing and Very Large Corpora. ACL.* USA (1999) 187–194

[8] Pedersen, T.: A simple approach to building ensembles of naive bayesian classifiers for word sense disambiguation. In *Proceedings of NAACL00.* USA (2000) 63–69

[9] Quinlan, J.R.: Induction of decision trees. In *Machine Learning* 1 (1986) 81–106.

[10] Rössler, M.: Using Markov Models for Named Entity recognition in German newspapers. In *Proceedings of the Workshop on Machine Learning Approaches in Computational Linguistics.* Italy (2002)29–37

[11] Tjong Kim Sang, E.F., Daelemans, W., Dejean, H., Koeling, R., Krymolowsky, Y., Punyakanok, V., Roth, D.: Applying system combination to base noun phrase identification. In *Proceedings of COLING00.* Germany (2000) 857–863

[12] Tjong Kim Sang, E.F.: Introduction to the CoNLL-2002 Shared Task: Language-Independent Named Entity Recognition. In *Proceedings of CoNLL-2002.* Taiwan (2002) 155–158

[13] Vapnik V. N.: *The Nature of Statistical Learning Theory.* Springer. (1995)

[14] Witten, I.H., Frank, E.: Data Mining. Machine Learning Algorithms in Java. Morgan Kaufmann Publishers (2000)

[15] Wu, D., Ngai, G., Carpuat, M.: A stacked, voted, stacked model for named entity recognition. In *CoNLL-2003. Computational Natural Language Learning.* Edmonton (2003) 200–203

Automatic Text Summarization with Genetic Algorithm-Based Attribute Selection

Carlos N. Silla Jr.[1], Gisele L. Pappa[2], Alex A. Freitas[2], and Celso A.A. Kaestner[1]

[1] Pontifícia Universidade Católica do Paraná (PUCPR),
Av. Imaculada Conceição 1155, 80215-901, Curitiba, PR, Brazil
{silla, kaestner}@ppgia.pucpr.br
[2] Computing Laboratory, University of Kent,
Canterbury, CT2 7NF, UK
{glp6, A.A.Freitas}@kent.ac.uk

Abstract. The task of automatic text summarization consists of generating a summary of the original text that allows the user to obtain the main pieces of information available in that text, but with a much shorter reading time. This is an increasingly important task in the current era of information overload, given the huge amount of text available in documents. In this paper the automatic text summarization is cast as a classification (supervised learning) problem, so that machine learning-oriented classification methods are used to produce summaries for documents based on a set of attributes describing those documents. The goal of the paper is to investigate the effectiveness of Genetic Algorithm (GA)-based attribute selection in improving the performance of classification algorithms solving the automatic text summarization task. Computational results are reported for experiments with a document base formed by news extracted from *The Wall Street Journal* of the TIPSTER collection–a collection that is often used as a benchmark in the text summarization literature.

1 Introduction

We are surely living in an era of information overload. Recent studies published by the University of Berkeley [8] indicate that in 2002 about 5 million *terabytes* of information were produced (in films, printed media or magnetic/optic storage media). This number is equivalent to twice as much the corresponding number for 1999, which indicates a growth rate of about 30% per annum. The *Web* alone contains about 170 *terabytes,* which is roughly 17 times the size of the printed material in the USA's Congress Library.

On the other hand, it is very difficult to use the available information. Many problems – such as the search for information sources, the retrieval/extraction of information and the automatic summarization of texts – became important research topics in Computer Science. The use of automatic tools for the treatment of information became essential to the user, because without those tools it is virtually impossible to exploit all the relevant information available in the *Web* [22].

In this scenario, the task of automatic text summarization is very important. The goal of an automatic text summarization system is to generate a summary of the original text that allows the user to obtain the main pieces of information available in that text, but with a much shorter reading time [12]. The summaries are produced based on attributes (or features) that are usually derived empirically, by using statistical and/or computational linguistics methods. The values of these attributes are derived from the original text, and the summaries typically have 10%-30% of the size of the original text [11].

One of the approaches that has been recently used to perform automatic text summarization is the use of Machine Learning methods [13]. In this context automatic text summarization is cast as a classification (supervised learning) task [5], [6], as will be discussed in Section 3. Other approaches for text summarization (which do not involve machine learning) are described in [11], [12].

In addition, an important data preprocessing task for effective classification is the attribute selection task, which consists of selecting the most relevant attributes for classification purposes [7]. This task is important because many original attributes can be irrelevant for classification, in which case their removal tends to improve the performance of the classification algorithm. Furthermore, attribute selection reduces the processing time taken by the classification algorithm, and it can also lead to the discovery of smaller, simpler classification models (e.g. smaller decision trees, as observed in this paper).

The goal of the paper is to investigate the effectiveness of Genetic Algorithm (GA)-based attribute selection in improving the performance of classification algorithms solving the automatic text summarization task. GAs have been chosen as the attribute selection methods because they have been very successful in this data preprocessing task [3], [2]. This is mainly due to their ability to cope well with attribute interaction (which is the crucial problem in attribute selection) [2]. More precisely, this paper investigates the effectiveness of two GAs for attribute selection in improving the performance of two different kinds of classification algorithms – viz. a decision tree-induction algorithm and the Naive Bayes classifier.

The remainder of this paper is organized as follows. Section 2 discusses GA-based attribute selection. Section 3 describes the *ClassSumm* system for summarization cast as a classification problem. Section 4 reports computational results. Finally, Section 5 presents the conclusions and discusses future work.

2 Attribute Selection with a Multi-objective Genetic Algorithm

Attribute selection is one of the most important tasks that precedes the application of data mining algorithms to real world databases [7]. It consists of selecting a subset of attributes relevant to the target data mining task, out of all original attributes. In this paper, the target task is classification, and one attribute is considered relevant if it is useful for discriminating examples belonging to different classes.

Attribute selection algorithms they differ from each other in two main components: the kind of search method they use to generate candidate attribute subsets and the way they evaluate the quality of a candidate attribute subset. The search methods can be classified in three main classes: exponential (e.g exhaustive search), randomised (e.g. genetic algorithms) and sequential (e. g. forward and backward sequential selection [7]) methods. In this paper we are interested in genetic algorithms (GA), since they are a robust search method, capable of effectively exploring large search spaces - which is usually the case in attribute selection. They also have the advantage of performing a global search – unlike many greedy, local search algorithms. In the context of data mining, this global search means that GAs tend to cope better with attribute interaction than greedy search methods [2].

The evaluation of the quality of each candidate solution can be based on two approaches: the filter and the wrapper approach. In essence, in the wrapper approach the attribute selection method uses the classification algorithm (as a black box) to evaluate the quality of a candidate attribute subset. In the filter approach the attribute selection method does not use the classification algorithm. We use the wrapper approach, because it tends to maximize predictive accuracy. (Note that this approach has the disadvantage of being significantly slower than the filter approach.)

The attribute selection task usually involves the optimisation of more than one objective, e.g. the predictive accuracy and the comprehensibility of the discovered knowledge. This is a challenging problem, because the objectives to be optimised can be conflicting with one another and they normally are non–commensurable – i.e., they measure different aspects of the target problem.

In the multi-objective optimisation framework [1], when many objectives are optimised there is no single best solution. Rather, there is a set of optimal solutions, each one involving a certain trade-off among the objectives. Multi-objective optimisation is based on Pareto dominance, that is: a solution $\mathbf{S}_1$ dominates another solution $\mathbf{S}_2$ iff $\mathbf{S}_1$ is not worse than $\mathbf{S}_2$ w.r.t. any objective and $\mathbf{S}_1$ is strictly better than $\mathbf{S}_2$ w.r.t. at least one objective. In multi-objective optimisation the system searches for non-dominated solutions.

MOGA is a Multi-Objective Genetic Algorithm designed to select attribute subsets for classification. It follows the basic ideas of GAs, i.e., it evolves a population of individuals, where each individual is a candidate solution to a given problem. In MOGA, each individual consists of M genes, where M is the number of original attributes in the data being mined. Each gene can assume values 0 or 1, indicating the absence or presence of the corresponding attribute in the selected subset of attributes.

Each individual is evaluated by a fitness function, which measures the quality of its attribute subset. At each generation (iteration) the fittest (the best) individuals of the current population survive and produce offspring resembling them, so that the population gradually contains fitter and fitter individuals – i.e., better and better candidate solutions to the underlying problem. The fitness function of MOGA is based on the wrapper approach, and involves the minimisation of both the classification error rate and the size of the decision tree built by J4.8[21]. MOGA searches for non-dominated solutions w.r.t. these two objectives. The version used in this paper returns, as the selected attribute subset, the non-dominated solution which dominates the largest number of solutions in the last generation. For more details about MOGA the reader is referred to [15],[16].

3 The ClassSumm System for Text Summarization

The *ClassSumm (Classification-based Summarizer)* system, proposed by [5], [6], is a system for automatic text summarization based on the idea of casting that task as a classification task and then using corresponding Machine Learning methods.

The system consists of the following main steps:

(1) the system extracts the individual sentences of the original documents, using one the approaches analysed in [18], in this work it was used the regular expression approach;
(2) each sentence is associated with a vector of predictor attributes (features), whose values are derived from the content of the sentence;
(3) each sentence is also associated with one of the following two classes: *Summary* (i.e., the sentence belongs to the summary) or *Not-Summary* (i.e., the sentence does not belong to the summary).

This procedure allows us to cast text summarization as a classification, supervised learning problem. As usual in the classification task, the goal of the classification algorithm is to discover, from the data, a relationship (say, an IF-THEN classification rule) that predicts the correct value of the class for each sentence based on the values of the predictor attribute for that sentence. More precisely, this casting leads to the following steps for solving a text summarization problem:

(1) The system constructs a training set where each example (record) corresponds to a sentence of the original documents, and each example is represented by a set of attribute values and a known class.
(2) A classification algorithm is trained to predict each sentence's class *(Summary* or *Not-Summary)* based on its attribute values.
(3) Given a new set of documents, the system produces a test set with predictor attributes in the same format as the training set. However, the values of the classes are unknown in the test set.
(4) Each sentence in the test set is classified, by the trained algorithm produced in step (2), in one of the two classes: *Summary* or *Not-Summary.*

Note that this procedure does not take into account the size of the summary to be generated. In practice the user often wants a summary of a specified size – in terms of percentage of the original document size. In order to take this into account, one uses a classification algorithm that, instead of directly predicting the class of each sentence, assigns to each sentence a measure of the relevance of that sentence for the summary. This produces a ranking of the sentences. Then the top *N* sentences in that ranking are assigned the class *Summary* and all the other sentences are assigned the class *Not-Summary,* where *N* is a user-specified parameter.

The classification algorithms used in the current version of ClassSumm are Naïve Bayes [13] and C4.5 [17]. In the former the relevance of a sentence for the summary is directly obtained from the conditional probability of the class *Summary* given the attribute values in the sentence. In the case of C4.5 the relevance of a sentence for the summary is obtained from the confidence factor associated with each leaf node of the induced tree.

The attributes used by ClassSumm can be categorized into two broad groups: shallow and deep attributes. Shallow attributes are based on heuristics and statistical methods; whereas deep attributes are based on linguistic knowledge. Both kinds of attributes are used in this paper. This work focuses on English texts only.

As usual in text processing systems, a preliminary preprocessing phase is performed [19]. This phase consists of four steps:

(1) identifying the sentences of the document; (2) converting all characters to lower case (*case folding*); (3) removing very common words *(stop words)* which do not contribute to the meaning of the text – e.g., "the", "a", etc.; (4) removing suffixes (i.e., performing *stemming*), so that words such as "learned" and "learning" are converted to the standard form "learn". These preprocessing steps help to significantly reduce the number of words, which is very important to improve the cost-effectiveness of automatic text summarization.

After this preprocesing, each sentence of the document is represented by an attribute vector consisting of the following elements:

1. Position: indicates the position of the sentence in the text, in terms of percentile, as proposed by Nevill-Manning [14];
2. Size: indicates the number of terms (words) in the sentence;
3. Average-TF-ISF: the TF-ISF (term frequency – inverse sentence frequency) measure [4] is a variation of the TF-IDF measure [20] widely used in information retrieval. (The difference between the two measures is explained in detail in [4].) The value of TF-ISF for each term of a sentence is computed, and the value of the Average-TS-ISF attribute for that sentence is the average value over the TF-ISF values for all the terms in that sentence;
4. Similarity to Title: The computation of this measure is based on the vectorial representation of the document, where each sentence is represented by a vector formed by its terms [20]. Initially the title of the document is preprocessed, forming a vector of terms, and then the similarity between each sentence and the title of the document is calculated by the co-sine measure [20];
5. Similarity to Keywords: Analogously to the previous attribute, this attribute is computed by using the vectorial representation of the document and calculating the similarity between each sentence and the vector of keywords by using the co-sine measure. This assumes the document has a set of author-provided keywords, which is the case in this work;
6. Cohesion w.r.t. All Other Sentences: This attribute is computed by calculating the distance between a sentence and every other sentence in the document. The sum of all those distances is the value of this attribute for the sentence in question;
7. Cohesion w.r.t. the Centroid: First the system computes the centroid of the document, which is simply a vector consisting of the arithmetic means of all sentence vectors' elements. Then the value of this attribute for a given sentence is computed by calculating the similarity between the sentence and the centroid vector – again, using the co-sine measure.

The next two attributes use a kind of linguistic structure built as an approximation to the text's rhetorical tree. This structure is obtained by running a hierarchical clustering algorithm, which forms clusters of similar sentences based on the vectorial representation of the sentences. The output of the clustering algorithm is a clustering tree where the leaf nodes are sentences and internal nodes represent clusters that have more and more sentences as the root of the tree is approached. The root of the tree represents a single cluster with all sentences in the document. The similarity measure used by the clustering algorithm is, again, the co-sine measure. Once a clustering tree has been produced by the hierarchical clustering algorithm, that tree is used to compute the following attributes for each sentence:

8. The depth of the sentence in the tree, i.e, the number of nodes that are ancestors of the leaf node representing that sentence.
9. The direction of the sentence in the tree, computed by following the path from the root towards the sentence up to depth four. At each depth level the direction can be *Left, Right* ou *None* (in case the current level is greater than the level of the sentence). This produces four attributes, each with one direction value. These attributes indicate the approximate position of the sentence in the rhetorical tree, incorporating linguistic knowledge into the set of predictor attributes.

The following attributes are obtained from the original text before the application of the preprocessing phase, and they also incorporate linguistic knowledge into the set of predictor attributes.

10. Indicators of Main Concepts: these indicators are computed by using a morphological *part-of-speech tagger* that identifies nouns in the document. The motivation for focusing on nouns is that they tend to be more meaningful (at least as individual words) than other part-of-speech classes. The 15 most frequent nouns in the document are selected to be the indicators of main concepts. For each sentence, the value of this attribute is true if the sentence contains at least one of those 15 indicators, and false otherwise.
11. Presence of Anaphors: From a linguistic point of view, the presence of anaphors in a sentence usually indicates that the information in the sentence is not essential, being used only to complement the information in a more relevant sentence. In *ClassSumm* the anaphors are identified by using a fixed list of words indicating anaphors. For each sentence, the value of this attribute is true if at least one of the first six words of the sentence is one of the words in the anaphor list, and false otherwise.
12. Presence of Proper Nouns: This attribute is computed directly from the output of a *part-of-speech tagger.* The value of the attribute is true if the sentence contains at least one proper noun, and false otherwise.
13. Presence of Discourse Markers: Some discourse markers, such as *because, furthermore,* also tend to indicate the presence of non-essential information. Discourse markers are identified by using a fixed list of words. The value of this attribute is true if the sentence contains at least one word in the list of discourse markers, and false otherwise.

Before the classification algorithm is applied to the training set, all the above non-binary attributes are normalized to the range [0..1] and then discretized. We adopt a simple "class-blind" discretization method, which consists of separating the original values into equal-width intervals; this procedure has produced good results in our previous experiments [5].

4 Computational Results

Previous work has reported results comparing *ClassSumm* with other Summarization methods [5], [6]. In those previous projects all original attributes were used. This paper focuses on a different issue. It investigates whether the performance of *ClassSumm* can be improved by using sophisticated attribute selection methods in a preprocessing step. An attribute selection method outputs only a subset of relevant attributes to be given to the classification algorithm, which hopefully will increase the predictive accuracy of the classification algorithm – which is also the accuracy of the decisions about which sentences should be included in the summary. The attribute selection methods used here are two kinds of Genetic Algorithms, namely a single-objective and a Multi-Objective Genetic Algorithm (MOGA) described in Section 2.

Experiments were carried out with a document base formed by news extracted from *The Wall Street Journal* of the TIPSTER collection [10]. This collection is often used as a benchmark in the text summarization literature.

For each document, a summary was produced using one of the following two approaches: (1) An automatically-generated summary, formed by the document's sentences that are most similar (according to the co-sine measure) to the summary provided by the author of the text, following the procedure proposed by Mani and Bloedorn [9]. This kind of summary is called an "ideal automatic summary". (2) A manually-generated summary, produced by an English teacher by selecting the most relevant sentences of the text. This is called an "ideal manual summary".

In all the experiments the training set consisted of 100 documents with their respective ideal automatic summaries. Experiments were carried out with two different kinds of test set. More precisely, in one experiment the test set consisted of 100 documents with their respective ideal automatic summaries, and in another experiment the test set consisted of 30 documents with their ideal manual summaries. In all experiments the training set and the test set were, of course, disjoint sets of documents, since the goal is to measure the predictive accuracy (generalisation ability) in the test set, containing only examples unseen during training.

In order to evaluate how effective Genetic Algorithm (GA)-based attribute selection is in improving the predictive accuracy of ClassSumm, two kinds of GAs for attribute selection have been used – both of them following the wrapper approach. The first one was the Multi-Objective GA (MOGA) discussed in Section 2. MOGA was used to select attributes for J4.8, a well-known decision-tree induction algorithm [21]. Recall that MOGA performs a multi-objective optimisation (in the Pareto sense) of both J4.8's error rate and the decision tree size. The results of training J4.8 with the attributes selected by the MOGA were compared with the results of training J4.8 with all original attributes, as a control experiment.

The second kind of GA used in the experiments was a simpler GA, called Single-Objetive GA (SOGA). It optimises only the error rate of a classification algorithm. SOGA was implemented directly from the MOGA implementation, by simply modifying MOGA's fitness function and selection method to optimise a single objective. Due to the focus on a single objective, the classifier used in this experiments was Naïve Bayes, whose measure of performance involves only error rate (no measure of size of the induced model). Again, the results of training Naïve Bayes with the attributes selected by the SOGA were compared with the results of training Naïve Bayes with all original attributes, as a control experiment.

The results are reported in Tables 1 and 2, which refer to the results for the test sets containing ideal automatic summaries and ideal manual summaries (as explained earlier), respectively. Each of these tables reports results for two kinds of summary size (10% and 20% of the original document). Finally, for each kind of test set and each summary size, the results of four methods are compared – two methods using the classification algorithms (J4.8 and Naïve Bayes) with all attributes and two methods using those algorithms with GA-based attribute selection, as explained above. In the experiments with J4.8 the reported results include the accuracy in the test set (in the range [0..1]), the decision tree size (number of tree nodes), and the number of selected attributes (or "all" – i.e., 16 attributes – when no attribute selection was done). In the experiments with Naïve Bayes, of course only the accuracy and number of selected attributes are reported.

Table 1. Results for test set containing "ideal" automatic summaries

Summary Size = 10% of original document			
Method	**Accuracy**	**Tree size**	**# selected attrib.**
J4.8	0.18	42	All
MOGA–J4.8	0.33	7	4
Naïve Bayes	0.39	N/a	All
SOGA–Naïve Bayes	0.38	N/a	9
Summary Size = 20% of original document			
Method	**Accuracy**	**Tree Size**	**# selected attrib.**
J4.8	0.44	164	All
MOGA–J4.8	0.47	4	2
Naïve Bayes	0.51	N/a	All
SOGA–Naïve Bayes	0.52	N/a	11

Table 2. Results for test set containing "ideal" manual summaries

Summary Size = 10% of original document			
Method	**Accuracy**	**Tree Size**	**# selected attrib.**
J4.8	0.15	42	All
MOGA–J4.8	0.25	7	3
Naïve Bayes	0.23	N/a	All
SOGA–Naïve Bayes	0.22	N/a	12
Summary Size = 20% of original document			
Method	**Accuracy**	**Tree Size**	**# selected attrib.**
J4.8	0.33	164	All
MOGA–J4.8	0.35	4	1
Naïve Bayes	0.36	N/a	All
SOGA–Naïve Bayes	0.35	N/a	11

Several trends in the results can be observed in Tables 1 and 2. First, as expected, in both Table 1 and Table 2 the accuracy associated with the larger summaries (20% of original document) is considerably larger than the accuracy associated with the smaller summaries (10% of the original document). This reflects the fact that, as the size of the summary increases, the classification problem becomes easier – e.g., the class distribution becomes less unbalanced (i.e, closer to a 50-50% class distribution).

Second, the GA-based attribute selection procedure had different effects on the performance of *ClassSumm,* depending on the kind of GA and classifier used in the experiments. The use of MOGA-based attribute selection led to an increase in J4.8's accuracy. This increase was very substantial in the smaller (10%) summaries, but relatively small in the larger (20%) summaries. In addition, MOGA-based attribute selection was very effective in selecting a very small number of attributes, which led to a very significant reduction in the size of the induced decision tree. This significantly improves the comprehensibility of discovered knowledge, an important goal in data mining [2], [21].

On the other hand, the effect of SOGA-based attribute selection in Naïve Bayes' accuracy was not so good. The effect was very small and, overall, even slightly negative.

These results for the two kinds of GA-based attribute selection are qualitatively similar in both Table 1 and Table 2, so that they are independent from whether the test set contains automatic summaries or manual summaries.

5 Conclusions and Future Work

As mentioned earlier, the goal of this paper was to investigate the effectiveness of Genetic Algorithm (GA)-based attribute selection in improving the performance of classification algorithms solving the automatic text summarization task. Overall, the two main conclusions of this investigation were as follows.

First, the Multi-Objective GA (MOGA) was quite effective. It led to an increase in the accuracy rate of the decision tree-induction algorithm used as a classifier, with a corresponding increase in the accuracy of the text summarization system. It also led to a very significant reduction in the size of the induced decision tree. Hence, the multi-objective component of the GA, which aims at optimising both accuracy and tree size, is working well.

Second, the Single-Objective GA (SOGA), which aimed at optimising classification accuracy only, was not effective. Surprisingly, there was no significant difference in the results of Naïve Bayes with all attributes and the results of Naïve Bayes with only the attributes selected by this GA. This indicates that all the original attributes seem more or less equally relevant for the Naïve Bayes classifier.

It should be noted that there is a lot of room for improvement in the results of the system, since the largest accuracy rate reported in Tables 1 and 2 was only 52%. Despite all the effort put into the design and computation of the 16 predictor attributes (which involve not only heuristic and statistical indicators, but also several relatively sophisticated linguistic concepts – e.g. a rhetorical tree), the current attribute set still seems to have a relatively limited predictive power. This suggests that future work could focus on designing an extended set of predictor attributes with more predictive power than the current one. Considering the difficult of doing this in a manual fashion, one interesting possibility is to use attribute *construction* methods to automatically create a better set of predictor attributes.

References

1. Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms. John Wiley & Sons (2001)
2. Freitas, A. A.: Data Mining and Knowledge Discovery with Evolutionary Algorithms. Springer (2002)

3. Kudo, M., Sklansky, J.: Comparison of algorithms that select features for pattern classifiers. Patten Recognition 33 (2000) 25–41
4. Larocca Neto, J., Santos, A.D., Kaestner, C.A.A., Freitas, A.A.: Document clustering and text summarization. In: Proc. 4th Int. Conf. Pratical Applications of Knowledge Discovery and Data Mining. (2000) 41–55
5. Larroca Neto, J., Freitas, A.A., Kaestner, C.A.A.: Automatic text summarization using a machine learning approach. In: XVI Brazilian Symposium on Artificial Intelligence. Number 2057 in Lecture Notes in Artificial Intelligence, Springer (2002) 205–215
6. Larroca Neto, J.: A Contribution to the Study of Automatic Text Summarization Techniques (in Portuguese). Master's thesis, Pontífica Universidade Católica do Paraná (PUC-PR), Graduate Program in Applied Computer Science. (2002)
7. Liu, H., Motoda, H.: Feature Selection for Knowledge Discovery and Data Mining. Kluwer Academic Publishers (1998)
8. Lyman, P., Varian, H.R.: How much information. (Retrieved from http://www.sims.berkeley.edu/ how-much-info-2003 on [01/19/2004]
9. Mani, I., Bloedorn, E.: Machine learning of generic and user-focused summarization. In: Proc. of the 15th National Conf. on Artificial Intelligence (AAI 98). (1998) 821–826
10. Mani, I., House, D., Klein, G., Hirschman, L., Obrsl, L., Firmin, T., Chrzanowski, M., Sundeheim, B.: The tipster summac text summarization evaluation. MITRE Technical Report MTR 92W0000138, The MITRE Corporation (1998)
11. Mani, I., Maybury, M.T.: Advances in Automatic Text Summarization. MIT Press (1999)
12. Mani, I.: Automatic Summarization. John Benjamins Publishing Company (2001)
13. Mitchell, T.M.: Machine Learning. McGraw-Hill (1997)
14. Nevill-Manning, C.G., Witten, I.H., Paynter, G.W., Frank, E., Gutwin, C.: KEA: Pratical Automatic Keyphrase Extraction. ACM DL 1999 (1999) 245–255
15. Pappa, G.L., Freitas, A.A., Kaestner, C.A.A.: Attribute selection with a multiobjective genetic algorithm. In: XVI Brazilian Symposium on Artificial Intelligence. Number 2057 in Lecture Notes in Artificial Intelligence, Springer (2002) 280–290
16. Pappa, G.L., Freitas, A.A., Kaestner, C.A.A.: A multi-objective genetic algorithm for attribute selecion. In: Proc. 4th Int. Conf. on Recent Advances in Soft Computing (RASC-2002), University of Nottingham, UK (2002) 116–121
17. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann (1993)
18. Silla Jr., C. N., Kaestner, C.A.A.: An Analysis of Sentence Boundary Detection Systems for English and Portuguese Documents. In: 5th International Conf. on Intelligent Text Processing and Computational Linguistics. Number 2945 in Lecture Notes in Computer Science, Springer (2004) 135–141
19. Sparck-Jones, K.: Automatic summarizing: factors and directions. In Mani, I.; Maybury, M. Advances in Automatic Text Summarization. The MIT Press (1999) 1–12
20. Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval. Information Processing and Management 24 (1988) 513–523
21. Witten, I.H., Frank, B.: Data Mining: Pratical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann, San Francisco (1999)
22. Zhong, N., Liu, J., Yao, Y.: In search of the wisdom web. IEEE Computer 35(1) (2002) 27–31

Coordination Revisited – A Constraint Handling Rule Approach

Dulce Aguilar-Solis and Veronica Dahl

Logic and Functional Programming Group,
Department of Computing Science,
Simon Fraser University,
Burnaby, B.C., Canada
{dma, veronica}@cs.sfu.ca

Abstract. Coordination in natural language (as in “Tom and Jerry”, “John built but Mary painted the cupboard”, “publish or perish”) is one of the most difficult computational linguistic problems. Not only can it affect any type of constituent, but it often involves “guessing” material left implicit. We introduce a CHR methodology for extending a user’s grammar not including coordination with a metagrammatical treatment of same category coordination. Our methodology relies on the input grammar describing its semantics compositionally. It involves reifying grammar symbols into arguments of a generic symbol `constituent`, and adding three more rules to the user’s grammar. These three rules can coordinate practically any kind of constituent while reconstructing any missing material at the appropriate point. With respect to previous work, this is powerfully laconic as well as surprisingly minimal in the transformations and overhead required.

1 Introduction

The present work forms part of our wider efforts to make spoken communication with computers a closer goal. We take inspiration from database theory (in particular, Datalog) and from constraint reasoning (in particular, CHRs) because the inherent ambiguity of natural language, and the frequency of errors that appear particularly in spoken or colloquial input, suggests bottom-up parsing techniques as the most promising. We should be able to gather, from an attempt to produce a correct parse, information which does not necessarily conform to strict grammar rules, and if possible, interpret that information in ways which allow us to somehow make sense of this “imperfect” input, just as humans do. Coordination adds one more level of difficulty, often involving implicit information that needs to be reconstructed from other conjoints (as in “The workshop was interesting and the talks, amusing”, in which the verb of the second conjoint (were) is implicit.

The advantages of bottom-up approaches to NL processing aimed at flexibility have been demonstrated within declarative paradigms for instance for parsing incorrect input and for detecting and correcting grammatical errors [2, 5, 6, 17].

Most of these approaches use constraint reasoning of some sort, sometimes blended with abductive criteria. For the specific problem of coordination in natural language, datalog approaches with constraints placed upon word boundaries have shown to be particularly adequate [11, 22].

CHRs [15] are of great interest to datalog inspired treatments of NL because, as shown in [6], the datalog part requires very little implementation machinery when using CHRGs: basically, grammar rules in CHRGs can be made to work either top-down or bottom-up according to the order in which their left and right hand sides are given, and CHRGs include an automatic insertion and handling of word boundaries [7, 9].

In this article we investigate the possibilities of blending CHRGs with datalog grammars, in view of a metagrammatical treatment of coordination. In our approach, the use of CHRGs together with a compositional meaning construction scheme is sufficient to understand (in the sense of machine understanding, i.e., to produce adequate meaning representations from) coordinated sentences, using a grammar which makes no explicit mention of coordination (other than declaring in its lexicons which are the allowed conjunctions). We choose a lambda-calculus based meaning representation to exemplify.

As in [2] or [5], we can build non-connected structures showing all partial successful analyses (but without having to replace trees by graphs as in that work). The result is a surprisingly simple approach, in which datalog grammars are given "for free" in the normal operation of CHRGs in conjunction with our transformation of the grammar given.

2 Background

2.1 Previous Approaches to Coordination

Typically, treatments of coordination involve either enriching the linguistic representation using explicit rules of grammar or adding special mechanism to the parsing algorithms to handle conjunction. The latter kind of approach has been used by Woods[23], Dahl and McCord[13], Haugeneder[16] and Milward[19]; among others.

In recent work ([11]), Dahl provided a left-corner plus charting datalog approach which can determine the parallel structures automatically, taking into account both syntactic and semantic criteria. This methodology records parse state constituents through linear assumptions to be consumed as the corresponding constituents materialize throughout the computation. Parsing state symbols corresponding to implicit structures remain as undischarged assumptions, rather than blocking the computation as they would if they were subgoals in a query. They can then be used to glean the meaning of elided structures, with the aid of parallel structures.

While being quite minimalistic in the amount of code required, this approach involves sophisticated process synchronization notions, as well as the use of linear affine implication. In the present work we show how to retain the advantages of [11] while drastically simplifying our methodology.

2.2 Constraint Handling Rules

CHR [15] are a committed-choice language for writing constraint solvers to be used with Prolog programs within domains such as real or integer numbers, as well as within less traditional domains, such as such as terminological and temporal reasoning. They are incorporated as an extension of (among others) SICStus Prolog[1].

As summarized in the CHR website[2], "CHR consist essentially of guarded rules that rewrite constraints into simpler ones until all the constraints have been solved. They define both simplification of and propagation over constraints. Simplification replaces constraints by simpler constraints while preserving logical equivalence (e.g. $X > Y, Y > X$ <=> fail). Propagation adds new constraints which are logically redundant but may cause further simplification (e.g. $X > Y, Y > Z ==> X > Z$). Repeatedly applying CHR incrementally simplifies and finally solves constraints (e.g. $A > B, B > C, C > A$ leads to fail)."

CHR works on constraint stores with its rules interpreted as rewrite rules over such stores. A string to be analyzed such as *"the sun shines"* is entered as a sequence of constraints {`token(0,1,the)`, `token(1,2,sun)`, `token(2,3,shines)`} that comprise an initial constraint store. The integer arguments represent word boundaries, and a grammar for this intended language can be expressed in CHR as follows.

```
token(X0,X1,the) ==> det(X0,X1,_).
token(X0,X1,sun) ==> n(X0,X1,sing).
token(X0,X1,shines) ==> v(X0,X1,sing).
n(X0,X1,Num), v(X1,X2,Num) ==> s(X0,X1,Num).
```

It is to be noted that if the application of a rule adds a constraint c to the store which already is there, no additional rules are triggered, e.g., $p ==> p$ does not loop as it is not applied in a state including p.

CHR applications to natural language processing include [1], which flexibly combines top-down and bottom-up computation in CHR, [6], which uses CHR to diagnose and correct grammatical errors, and [10], which implements property grammars using CHR.

2.3 Constraint Handling Rule Grammars

CHRGs, or Constraint Handling Rule Grammars, were developed by H. Christiansen[3] as a grammatical counterpart for CHRs. A *CHRG* consists of finite sets of *grammar* and *constraints symbols* and a finite set of *grammar rules.* A *grammar symbol,* is formed as a logical atom whose predicate symbol is a grammar symbol; a grammar symbol formed by `token/1` is called a *terminal,* any other grammar symbol a *nonterminal.* A *propagation (grammar) rule* is of the form

$$\alpha\beta\gamma ==> G|\delta.$$

[1] http://www.sics.se/isl/sicstuswww/site/documentation.html

[2] http://www.informatik.uni-ulm.de/pm/mitarbeiter/fruehwirth/chr.html

[3] http://www.dat.ruc.dk/ henning/chrg/

The part of the rule preceding the arrow is called the *head*, G the *guard*, and δ the body; $\alpha, \beta, \gamma, \delta$ are sequences of grammar symbols and constraints so that β contains at least one grammar symbol, and δ contains exactly one grammar symbol which is a nonterminal (and perhaps constraints); α (γ) is called *left (right) context* and β the *core* of the head; G is a guard as in CHR that may test properties for the variables in the head of the rule. If either the left or the right context is empty, the corresponding marker is left out and if G is empty (interpreted as `true`), the vertical bar is left out. The convention from DCG is adopted that non-grammatical constraints in head and body of a rule are enclosed in curly brackets.

CHRG rules can be combined with rules of CHR and with Prolog, which is convenient for defining the behaviour of non-grammatical constraints. CHRG includes also notation for gaps and for parallel matching, which we neither describe nor use in the present work.

The CHRG notation makes the word boundary arguments implicit and, analogously to DCGs, includes a syntax for using nongrammatical constraints.

As demonstrated by [7, 8], CHRG is a very powerful system which provides straightforward implementation of assumption grammars [12], abductive language interpretation and a flexible way to handle interesting linguistic phenomena such as anaphora and a restricted form of coordination (specifically, for coordinating subject and object proper names). This work motivated us to try virtual coordination through CHRGs at the general level.

3 Our Approach to Coordination

3.1 General Description

Our methodology imposes two requirements on the user's grammar. First, semantics must be defined compositionally. Second, semantic material must be isolated into one specific argument, so that the rules for virtual coordination can easily identify and process it. For instance, if we use the second argument for semantics, a rule such as

```
  name(X) -> np(X^Sem^Sem).
```

should be coded in CHR and CHrg as

```
CHR:   category(name,X,P0,P1) ==> constituent(np,X^Sem^Sem,P0,P1).
CHRG:  category(name,X) ::> constituent(np,X^Sem^Sem).
```

We assume that there are two (implicit or explicit) coordinating constituents, C1 and C2, surrounding the conjunction, which must in general be of the same category[4]. As in Dahl's previous work [11], we adopt the heuristics that closer scoped coordinations will be attempted before larger scoped ones. Thus in Woods' well-known example[23], *"John drove his car through and demolished a window"*, "`vp conj vp`" is tried before "`sent conj sent`".

[4] This is a simplifying assumption: coordination can involve different categories as well, but in this paper we only address same category coordination.

If both C1 and C2 are explicit, we simply postulate another constituent of the same category covering both, and conjoin their meanings. This is achieved through the rule:

```
    constituent(C,Sem1,P0,P1),
    constituent(conj,_,P1,P2),
    constituent(C,Sem2,P2,P3) ==>
                                constituent(C,Sem,P0,P3),conj(Sem1,Sem2,Sem).
```

Sem can take either the form and(Sem1,Sem2) or a more complex form.

If either C1 or C2 is implicit, the CHRG engine will derive all possible partial analyses and stop with no complete sentence having been parsed. We can then resume the process after dynamically adding the following symmetric rules, in charge of completing the target in parallel with the source. Once the target has been completed, the above rule for coordinating complete constituents can take over.

```
% The second conjoint is incomplete
    constituent(C,Sem1,P0,P1),
    constituent(conj,_,P1,P2) ==>
                                complete(C,Sem1,range(P0,P1),Sem2,P3)
                            |   constituent(C,Sem2,P2,P3).
% The first conjoint is incomplete
    constituent(conj,_,P1,P2),
    constituent(C,Sem2,P2,P3) ==>
                                complete(C,Sem2,range(P2,P3),Sem1,P0).
                            |   constituent(C,Sem1,P0,P1).
```

complete/5 generates the features of a constituent of category C that is incomplete between the given points, using the source (the portion of text indicated by range/2) as parallel structure. The new constraint is left in the constraint store, so that the rule for complete conjoint coordination can apply.

3.2 An Example

For *Philippe likes salsa and Stephane tango,* we have the initial constraint store: {philippe(0,1),likes(1,2),salsa(2,3),and(3,4),stephane(4,5),tango(5,6)} to which the following "constraints" (in the sense of the CHR store) are successively added: [name(0,1),verb(1,2),noun(2,3),conj(3,4),name(4,5),noun(5,6), np(0,1),vp(1,3),np(2,3),np(4,5),np(5,6),s(0,3)].

At this point, since the analysis of the entire sentence is not complete, the dynamically added rules will compare the data to the left and right of the conjunction, noting the parallelisms present and absent:

```
np(0,1) parallel to np(4,5)
verb(1,2) parallel to ?
np(2,3) parallel to np(5,6)
```

and postulate a verb(5,5), with the same surface form ("likes") as the verb in the parallel structure. The addition of this element to the constraint store triggers the further inferences: {`vp(5,6), s(4,6)`}. This in turn will trigger the complete constituent coordination rule, resulting in {`s(0,6)`}.

3.3 Top-Down Prediction

For cases in which the two parallel structures are different, the simple pairing of constituents in the source and target, as above, will not be enough. For Woods' example, *"John drove a car through and demolished a window"*, we have the constraint store:

```
{ john(0,1), drove(1,2), a(2,3), car(3,4),
  through(4,5), and(5,6), demolished(6,7),
  a(7,8), window(8,9), name(0,1), verb(1,2),
  det(2,3), noun(3,4), prep(4,5), conj(5,6),
  verb(6,7), det(7,8), noun(8,9), np(0,1),
  np(7,9), vp(6,9) }
```

In such cases we examine the grammar rules in top-down fashion to determine which ones would warrant the completion of a constituent that appears to one side of the conjunction but not to the other.

In our example, the candidate sources are: prep(4,5), verb(6,7) and vp(6,9). Postulating a missing prep at point 6 or a missing verb at point 5 does not lead to success, so we postulate a vp ending in point 5 and use vp rules top-down to predict any missing constituents. The pp rule is eventually found, which rewrites pp into a preposition plus a noun phrase, so we can postulate a missing noun phrase between point 5 and itself, to be filled in by the parallel noun phrase in the source, namely *"a window"*(we must avoid requantification, so the window driven through and the one demolished must be the same window). This new noun phrase triggers in turn the inference of a verb phrase between points 1 and 5, which in turn allows us to conjoin the two verb phrases, and complete the analysis.

3.4 Semantics

We now turn our attention to grammars which construct meaning representations for the sentences analysed.

After having determined the parallel elements in source and target, we must void the source's meaning (using for instance higher-order unification) from those elements which are not shared with the target, and only then apply the resulting property on the representation of the target.

For our example, we must from the meaning representation of *"Philippe likes salsa"* reconstruct the more abstract property[$\lambda y.\lambda x.$likes(x,y)], which can then be applied on *"tango"* and *"stephane"* to yield likes(stephane,tango).

We bypass this need by keeping copies of the abstract properties as we go along. While the original properties get instantiated during the analysis (so that the meaning of *"likes"* in the context *"Philippe likes salsa"* becomes `likes(philippe,salsa)`), their copies do not. It is these uninstantiated copies that are used as meanings for the reconstructed targets.

3.5 Syntactic Considerations

Some of the syntactic features carried around by typical grammars need a special treatment by the coordination rules: the conjunction of two singular noun phrases (e.g. *"the cat and the dog"*), for instance, should result in a plural conjoined noun phrase. Our system takes this into account.

4 Related and FutureWork

Various authors, e.g.[18] discuss an alternative approach to anaphoric dependencies in ellipsis, in which the dependence between missing elements in a target clause and explicit elements in a source clause does not follow from some uniform relation between the two clauses, but follows indirectly from independently motivated discourse principles governing pronominal reference. While containing linguistically deep discussions, the literature on this discourse-determined analysis also focusses on ellipsis resolution, while still leaving unresolved (to the best of our knowledge) the problem of automatically determining which are the parallel structures.

On another line of research, Steedman's CCGs [21] provide an elegant treatment of a wide range of syntactic phenomena, including coordination, which does not resort to the notions of movement and empty categories, instead using limited combinatory rules such as type raising and functional composition . However, these are also well known to increase the complexity of parsing, originating spurious ambiguity- that is, the production of many irrelevant syntactic analyses as well as the relevant ones. Extra work for getting rid of such ambiguity seems to be needed, e.g. as proposed in [20].

The results in [14] concerning the use of the distinction between primary and secondary occurrences of parallel elements in order to provide ambiguous readings of discourses such as *"Jessie likes her brother. So does Hannah."* could, in principle, be transferred into our approach as well.

A recent line of work in Computational Linguistics - the property based paradigm[3, 4]- departs from the Chomskyan, hierarchical approach to parsing in that it accepts input through checking properties between any two words or constituents modularly, and relaxing some of the properties to accomodate ill-formed or incomplete input. Properties are statically declared as either necessary or desirable. In this approach, a sentence such as *"The workshops was interesting"* may be accepted, while producing an indication that the property of number agreement between the noun phrase and the verb is not satisfied (as we might want for instance in a language tutoring system). Thus it is not necessary to have a complete parse tree to get interesting results: incomplete or incorrect input will yield partial results, and the users' static indications of which properties can be relaxed and which should be enforced will result in indications by the system of which properties are not satisfied for a given input.

A CHR methodology for parsing in the property based paradigm has been shown in [10], which proceeds from head words and projects them into phrases (e.g., a noun can form a noun phrase all by itself). Then it tries to incorporate more constituents into this phrase, one at a time, by testing properties between the constructed phrase and the potential addition. This results in basically a one-rule parser, whose single rule successively takes two categories, checks the properties between them, and constructs a new category. This process repeats until no more new categories can be inferred. We are currently working on extending this parser with coordinating abilities as described in this paper, both as a further proof of concept of the ideas here developed, and as an interesting approach to treating long distance dependencies in property based grammars.

5 Concluding Remarks

We have shown how to take advantage of CHRGs' built-in datalog facilities and constraint store management in order to treat metagrammatical coordination in a relatively simple while effective and encompassing manner.

A few observations are in order: as we have seen, a simple conjoining of the representations obtained for the parallel structures as proposed in [14] may not suffice. Since these structures may be quite dissimilar, we must conjoin only the parallel elements. We postulate that, in compositionally defined semantics, the parallel elements will be represented by those subterms which are not unifiable.

Another important observation is that we do not need to commit to higher-order unification, property reconstructions, etc. Again for compositionally defined semantics, the parallel structures analysis together with top-down target determination ensures that the correct meanings are associated to the elided constituents as a side effect of parsing.

Lastly, we should note that our analysis allows for the source clause to not necessarily be the first one- as we have also seen, we can have structures in which the incomplete substructure does not antecede the complete one. Thus our analysis can handle more cases than those in the previous related work.

Acknowledgements

This research was made possible by Veronica Dahl's NSERC research grant. We thank the anonymous referees for their useful comments on a previous draft of this paper.

References

1. Abdennadher, S. and Schtz, H. CHR: A Flexible Query Language. In International conference on Flexible Query Answering Systems, FQAS'98, LNCS, Springer, Roskilde, Denmark (1998)
2. Balsa, J., Dahl, V. and and Pereira Lopes, J. G. Datalog Grammars for Abductive Syntactic Error Diagnosis and Repair. In Proceedings of the Natural Language Understanding and Logic Programming Workshop, Lisbon (1995)
3. Bès G. and Blache, P. Propriétés et analyse d'un langage. In Proceedings of TALN'99 (1999)

4. Bès G., Blache, P., and Hagège, C. The 5P Paradigm. Research report, GRIL/LPL (1999)
5. Blache, P. and Azulay, D. Parsing Ill-formed Inputs with Constraints Graphs. In A. Gelbukh (ed), Intelligent Text Processing and Computational Linguistics, LNCS, Springer, 220–229 (2002)
6. Christiansen, H. and Dahl, V. Logic Grammars for Diagnosis and Repair. In Proceedings of 14th IEEE International Conference on Tools with Artificial Intelligence, Washington D.C., 307–314 (2002)
7. Christiansen, H. Logical Grammars Based on Constraint Handling Rules,(Poster abstract). In Proc. 18th International Conference on Logic Programming, Lecture Notes in Computer Science, 2401, Springer-Verlang, p. 481 (2002)
8. Christiansen, H. Abductive Language Interpretation as Bottom-up Deduction. In Proc. NLULP 2002, Natural Language Understanding and Logic Programming, Wintner, S. (ed.), Copenhagen, Denmark, 33–48 (2002)
9. Christiansen, H. CHR as Grammar Formalism, a First Report. In Sixth Annual Workshop of the ERCIM Working Group on Constraints, Prague (2001)
10. Dahl, V. and Blache, P. Directly Executable Constraint Based Grammars. In Proc. Journees Francophones de Programmation en Logique avec Contraintes, Angers, France (2004).
11. Dahl, V. On Implicit Meanings. In Computational Logic: from Logic Programming into the Future. F. Sadri and T. Kakas (eds), Springer-Verlang (2002)
12. Dahl, V., Tarau, P., and Li, R. Assumption Grammars for Processing Natural Language. In Fourteenth International Conference on Logic Programming, MIT Press, 256–270 (1997)
13. Dahl, V. and McCord, M. Treating Coordination in Logic Grammars. In American Journal of Computational Linguistics 9, 69–91 (1983)
14. Darlymple, M., Shieber, S., and Pereira, F. Ellipsis and Higher-Order Unification. In Linguistics and Philosophy, 14(4), 399–452 (1991)
15. Fruhwirth, T. W. Theory and Practice of Constraint Handling Rules. In Journal of Logic Programming, 37, 95–138 (1998)
16. Haugeneder, H. A Computational Model for Processing Coordinate Structures: Parsing Coordination will-out Grammatical Specification. In ECAI 1992, 513–517 (1992)
17. Kakas, A.C., Michael, A., and Mourlas, C. ACLP: Abductive Constraint Logic Programming. The Journal of Logic Programming, Vol.44, pp. 129–177, 2000.
18. Kehler, A. and Shieber, S. Anaphoric Dependencies in Ellipsis. In Computational Linguistics, 23(3) (1997)
19. Milward, D. Non-Constituent Coordination: Theory and Practice In Proceedings of COLING 94 , Kyoto, Japan, 935-941 (1994)
20. Park, J.C and Cho, H.J. Informed Parsing for Coordination with Combinatory Categorial Grammar. COLING'00, pp. 593–599, (2000)
21. Steedman, M. Gapping as Constituent Coordination. In Linguistics and Philosophy (1990)
22. Voll, K., Yeh, T., and Dahl, V. An Assumptive Logic Programming Methodology for Parsing. In International Journal on Artificial Intelligence Tools (2001)
23. Woods, W. An Experimental Parsing System for Translation Network Grammars. In R. Rustin, editor, Natural Language Processing, Algorithmic Press, New York, 145–149 (1973)

Appendix: A Sample Grammar

```
:- compile(chrg).
handler parser.
grammar_symbols c/3.

c(C, F1, X),c(con,[],Con),c(C,F2,Y) ::>
           conjunction_agree(Con,C,F1,F2,F), collapse(C,X,Y,Z)
     |     c(C,F, Z).
% The second conjoint is incomplete
c(C,T,Sem1):(P0,P1), c(con,_,_):(P1,P2) ::>
           completar(C,T,Sem1,range(P0,P1),Sem2,P3,Ts)
     |     c(C,Ts,Sem2):(P2,P3).
% The first conjoint is incomplete
c(con,_,_):(P1,P2), c(C,T,Sem1):(P2,P3),  ::>
           completar(C,T,Sem1,range(P2,P3),Sem2,P0,Ts)
     |     c(C,Ts,Sem2):(P0,P1).

c(np,[num:N],X^Sco^Sem), c(vp,[ten:T1,num:N1],X1^Sco1) ::>
           apply((N=N1),N,NewN), apply((X=X1,Sco=Sco1),Sem,NewSem)
     |     c(s,[num:NewN],NewSem).
c(tv,[ten:T,num:N],Sem),c(np,[num:_],Y1^Sco1^Pred),c(pp,Prep,NP)::>
           add_pp(Prep,Sem,NP,Y^X^Sco),
           apply((Y=Y1,Sco=Sco1),X^Pred,NewSem)
     |     c(vp,[ten:T,num:N],NewSem).
c(tv,[ten:T,num:N],Y^X^Sco),c(np,[num: _],Y1^Sco1^Pred) ::>
           apply((Y=Y1,Sco=Sco1),X^Pred,NewSem)
     |     c(vp,[ten:T,num:N],NewSem).
c(iv,F,Sem), c(pp,Prep,NP) ::>
           add_pp(Prep,Sem,NP,NewSem)
     |     c(vp,F,NewSem).
c(iv,F,Sem) ::> c(vp,F,Sem).
c(name,[num: N],X) ::> c(np,[num: N],X^Sem^Sem).
c(det,[num: N],X^Res^Scope^Sem),c(noun,[num:N1],X1^Res1) ::>
           apply((N=N1),N,NewN),
           apply((X=X1,Res=Res1),X^Scope^Sem,NPsemanticTerm)
     |     c(np,[num:NewN],NPsemanticTerm).
c(prep,_,Prep), c(np,_,NP) ::> c(pp,Prep,NP).
```

Question Answering for Spanish Based on Lexical and Context Annotation

Manuel Pérez-Coutiño, Thamar Solorio, Manuel Montes-y-Gómez[†], Aurelio López-López, and Luis Villaseñor-Pineda

Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE),
Luis Enrique Erro No. 1, Sta Ma Tonantzintla, 72840, Puebla, Pue, México
{mapco, thamy, mmontesg, allopez, villasen}@inaoep.mx

Abstract. Question Answering has become a promising research field whose aim is to provide more natural access to the information than traditional document retrieval techniques. In this work, an approach centered in the use of context at a lexical level has been followed in order to identify possible answers to short factoid questions stated by the user in natural language. The methods applied at different stages of the system as well as an architecture for question answering are described. The evaluation of this approach was made following QA@CLEF03 criteria on a corpus of over 200,000 news in Spanish. The paper shows and discusses the results achieved by the system.

Keywords: Question Answering, Automatic Text Processing, Natural Language Processing.

1 Introduction

Question Answering (QA) systems has become an alternative to traditional information retrieval systems because of its capability to provide concise answers to questions stated by the user in natural language. This fact, along with the inclusion of QA evaluation as part of the Text Retrieval Conference (TREC)[1] in 1999, and recently [6] in Multilingual Question Answering as part of the Cross Language Evaluation Forum (CLEF)[2], have arisen a promising and increasing research field.

Nowadays, the state of the art on QA systems is focused in the resolution of factual questions [2, 14] that require a named entity (date, quantity, proper noun, locality, etc) as response. For instance, the question *"¿Cuándo decidió Naciones Unidas imponer el embargo sobre Irak?"*[3] demands as answer a date *"en agosto de 1990"*[4]. Several approaches of QA systems like [8, 13, 4, 10] use named entities at different stages of

† This work was done while visiting the Dept. of Information Systems and Computation Polytechnic University of Valencia, Spain.

1 http://trec.nist.gov/

2 http://clef-qa.itc.it/

3 When did the United Nations decide to impose the embargo on Iraq?

4 In August 1990.

the system in order to find a candidate answer. Generally speaking, the use of named entities is performed at the final stages of the system, i.e., either in the passage selection or as a discriminator in order to select a candidate answer at the final stage. Another interesting approach is the use of *Predictive Annotation* which was first presented at TREC-8 by Prager et al. [8]. One meaningful characteristic of this approach is the indexing of anticipated semantic types, identifying the semantic type of the answer sought by the question, and extracting the best matching entity in candidate answer passages. In their approach, the authors used no more than simple pattern matching to get the entities. The system described in this document was developed to process both, questions and source documents in Spanish. Our system is based on approach just described but differs in the following: i) the identification of the semantic classes relies in the preprocessing of the whole document collection by a POS tagger that simultaneously works as named entity recognizer and classifier. ii) the indexing stage takes as item the lexical context associated to each single named entity contained in every document of the collection. iii) the searching stage selects as candidate answers those named entities whose lexical contexts match better the context of the question. iv) at the final stage, candidate answers are compared against a second set of candidates gathered from the Internet. v) Final answers are selected based on a set of relevance measures which encompass all the information collected in the searching process. The evaluation of the system was made following the methodology and data set of QA@CLEF-2003 [6] in order to get a comparable evaluation with other systems designed for Spanish language.

The rest of this paper is organized as follows; section two describes the architecture and functionality of the system; section three details the process of question processing; section four details the process of indexing; section five shows the process of searching; section six describe the process of answer selection; section seven discusses the results achieved by the system; and finally section eight exposes our conclusions and discusses further work.

2 System Overview

The system adjusts to a typical QA system architecture [14]. Figure 1 shows the main blocks of the system. The system could be divided into the following stages: *question processing,* which involves the extraction of named entities and lexical context in the question, as well as question classification to define the semantic class of the answer expected to respond to the question; *indexing,* where a preprocessing of the supporting document collection is done, building the representation of each document that become the searching space to find candidate answers to the question; *searching,* where a set of candidate answers is obtained from the index and the Internet, (here candidate answers are classified by a machine learning algorithm, and provides information to perform different weighting schemes); and finally *answer selection* where candidate answers are ranked and the final answer recommendation of the system is produced. Next sections describe each of these stages.

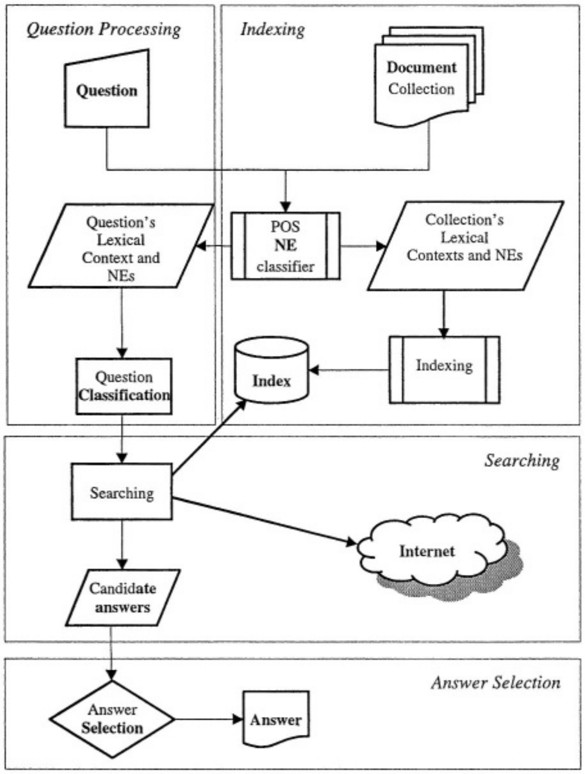


Fig. 1. Block diagram of the system. There are four stages: question processing, indexing, searching and answer selection

3 Question Processing

MACO [3] is a POS tagger and lemmatizer capable of recognizing and classifying named entities (NEs). The possible categories for NEs are the following: person, organization, geographic place, date, quantity and miscellaneous. In order to reduce the possible candidate answers provided by our system we perform a question classification process. The purpose of this classification is to match each question with one of the six named entities provided by MACO.

We use a straightforward approach, where the attributes for the learning task are the prefixes of the words in the question and additional information acquired by an Internet search engine.

The procedure for gathering this information from Internet is first we use a set of heuristics in order to extract from the question the first noun word or words *w*. We then employ a search engine, in this case Google, submitting queries using the word *w* in combination with the five possible semantic classes. For instance, for the question *Who is the President ofthe French Republic?* President is extracted as the noun in the question using our heuristics, and run 5 queries in the search engine, one for each possible class. The queries take the following forms:

- “President is a person”
- “President is a place”
- “President is a date”
- “President is a measure”
- “President is an organization”

For each query (q_i) the heuristic takes the number of results (Cr_i) returned by Google and normalizes them according to equation 1. This means that for each question, the summatory of their five performed queries is 1. Normalized values $(Iw(q_i))$ are taken as attributes values for the learning algorithm. As it can be seen is a very direct approach, but experimental evaluations showed that this information gathered from Internet is quite useful [11].

The machine learning technique used was Support Vector Machines [12] implemented in WEKA [15].

$$Iw(q_i) = Cr_i \Big/ \sum_{i=0}^{n} Cr_i \quad \text{Equation 1.}$$

4 Indexing

Each document in the collection is modeled by the system as a factual text object whose content refers to several named entities even when it is focused on a central topic. As mentioned, named entities could be one of these objects: persons, organizations, locations, dates and quantities. The model assumes that the named entities are strongly related to their lexical context, especially to nouns (subjects) and verbs (actions). Thus, a document can be seen as a set of entities and their contexts. For details about the document model we refer the reader to [7]. In order to obtain the representation of the documents, the system begins preprocessing each document with MACO, where this process is performed off-line. Once the document collection has been tagged, the system extracts the lexical contexts associated to named entities. The context considered for this experiment consists of the four verbs or nouns, both at the left and right of its corresponding NE. The final step in the indexing stage is the storage of the extracted contexts, populating a relational database[5] which preserves several relations between each named entity, its semantic class, associated contexts, and the documents where they appeared. In other words, the index is an adaptation of the well knows inverted file structure used in several information retrieval systems. Given the information required by the system, the indexing and searching modules were developed from scratch.

5 Searching

The search engine developed for the system and the searching process differ in several aspects from traditional search engines. This process relies on two information sources: first the information gathered from question processing, i.e., the expected

[5] Due to performance constraints, the index has been distributed over a cluster of 5 CPUs.

semantic class of the answer to the question, and the named entities and lexical context of the question; and second, the index of named entities, contexts and documents created during indexing.

5.1 Searching Algorithm

With the document representation, all the name entities mentioned in a given document can be known beforehand. Thus, the name entities from the question become key elements in order to define the document set more likely to provide the answer. For instance, in the question *"¿Cuál es el nombre del presidente de México?"*[6], the named entity "Mexico" narrows the set of documents to only those containing such name entity. At the same time, another assumption is that the context in the neighborhood of the answer has to be similar to the lexical context of the question. Once more, from the question of the example, the fragment "even before his inauguration as president of Mexico, Vicente Fox…" contains a lexical context next to the answer which is similar to that of the question.

Following is the algorithm in detail:

1. Identify the set of relevant documents according to the named entities in the question.
2. Retrieve all contexts in each relevant document.
3. Compute the similarity between question context and those obtained in step 2.
   - 3.1. Preserve only those contexts whose associated named entity corresponds to the semantic class of the question.
   - 3.2. Compute a similarity function based on frequencies to perform further ranking and answer selection.
4. Rank the candidate named entities in decreasing order of similarity.
5. Store similarity and named entity classification information (step 3.2) for next stage.

6 Answer Selection

Analyzing the output from the local index we find out that we had a lot of possible answers with the same values for similarity and named entity classification information. Thus, we develop a method for selecting the final possible answer based on answers retrieved from Internet and automated classification of answers using a bagged ensemble of J48 [15].

The final answer presented by our system was selected by calculating the intersection among words between the local index candidate answers and the answers provided by the Internet search. We consider the candidate answer with highest intersection value to be more likely to be the correct answer. However, in some cases all the candidate answers have the same intersection values. In this case we selected from the candidates the first one classified by the learning algorithm as belonging to the positive class. When no positive answer was found among the candidates for a question, then we selected the first candidate answer with highest value from the local index.

[6] What is the name of the president of Mexico?

The following sections briefly describe the Internet search and the answer classification processes.

6.1 Internet Searching

As mention earlier, at the final stage, the system uses information from the Internet in order to get more evidence of the possible accuracy of each candidate answer. From the perspective of the overall system, Internet searching occurs simultaneously to the local search. This subsection reviews the process involved in such task.

The module used at this step was originally developed at our laboratory to research the effectiveness of a statistical approach to web question answering in Spanish. Such approach lies in the concept of redundancy in the web, i.e, the module applies a several transformations in order to convert the question into a typical query and then this query along to some query reformulations are sent to a search engine with the hypothesis that the answer would be contained –several times– in the snippets retrieved by the search engine[7]. The selection of candidate answers from Internet is based on computing all the n-grams, from unigrams to pentagrams, as possible answers to the given question. Then, using some statistical criteria the n-grams are ranked by decreasing likelihood of being the correct answer. The top ten are used to validate the candidates gathered from the local searching process.

6.2 Answer Classification

Discriminating among possible answers was posed as a learning problem. Our goal was to train a learning algorithm capable of selecting from a set of possible candidates the answer that most likely satisfies the question. We selected as features the values computed by the local indexing. We use five attributes: 1) the number of times the possible answer was labeled as the entity class of the question; 2) the number of times the possible entity appeared labeled as a different entity class; 3) number of words in common in the context of the possible answer and the context of the question, excluding named entities; 4) the number of entities that matched the entities in the question, and 5) the frequency of the possible answer along the whole collection of documents. With these attributes, we then trained a bagged ensemble of classifiers using as base learning algorithm the rule induction algorithm J48 [9].

In this work we build the ensemble using the bagging technique which consists of manipulating the training set [1].

Given that we had available only one small set of questions, we evaluate the classification process in two parts. We divided the set of questions into two subgroups of the same size and performed two runs. In each run, we trained on one half and tested on the other.

7 System Evaluation

The evaluation of the system was made following the methodology used in the past QA track at CLEF-2003 [6]. Following, the criteria used in this track is summarized.

[7] The search engine used by this module is Google (http://www.google.com).

The document collection used was EFE94, provided by the Spanish news agency EFE. The collection contains a total of 215,738 documents (509 MB). The question set is formed by 200 questions; and 20 have no answer in the document set. For such questions the system has to answer with the string NIL. Answers were judged to be incorrect (W) when the answer-string did not contain the answer or when the answer was not responsive. In contrast, a response was considered to be correct (R) when the answer string consisted of nothing more than the exact, minimal answer and when the document returned supported the response. Unsupported answers (U) were correct but it was impossible to infer that they were responsive from the retrieved document. Answers were judged as non-exact (X) when the answer was correct and supported by the document, but the answer string missed bits of the response or contained more than just the exact answer. In strict evaluation, only correct answers (R) scored points, while in lenient evaluation the unsupported responses (U) were considered to be correct, too.

The score of each question was the reciprocal of the rank for the first answer to be judged correct (1 or 0, or 0.333, or 0.5 points), depending on the confidence ranking. The basic evaluation measure is the Mean Reciprocal Rank (MRR) that represents the mean score over all questions. MRR takes into consideration both recall and precision of the systems' performance, and can range between 0 (no correct responses) and 1 (all the 200 queries have a correct answer at position one).

7.1 Results

Table 1 shows the results gathered by our system, the total of questions correctly answered is 85, which represents a 42.5% of the question set. It is important to remark that 87% of the answers are given as first candidate for the system.

Table 1. Results gathered from the system after processing the QA@CLEF-2003 question set

Rank	1st	2nd	3rd
Number of correct answers	74	9	2
Total of correct answers	85 (42.5%)		
Mean Reciprocal Rank	0.3958		

Table 2 shows the comparative results between the best run (Alicex031ms) presented last year in the QA monolingual task for Spanish [13] and the results gathered by our system in this work (Inaoe).

Table 2. Results from QA@CLEF-2003 monolingual task and our system

	Strict		Lenient	
Run	MRR	Correct	MRR	Correct
Alicex031ms	0.3075	40.0 %	0.3208	43.5 %
Inaoe	0.3958	42.5%	---	---

Given the approach followed by our system it is unable to evaluate it under lenient parameters, i.e, the systems provides as answers named entities avoiding non-exact (X) or unsupported (U) answers. However the MRR achieved by our approach is higher than both strict and lenient MRR of Alicex031ms.

8 Conclusions

This work has presented a lexical-context approach for QA in Spanish. Such approach has been evaluated on a standard test bed and demonstrated its functionality. The strength of this work lies in the model used for the source documents. The identification and annotation in advance of named entities and their associated contexts serves as key information in order to select possible answers to a given factoid question. On the other hand, the discrimination of candidate answers is a complex task that requires more research and experimentation of different methods. In this work we have experimented with the merging of evidence coming from three main sources: a ranked list of candidate answers gathered by a similarity measure, answer classification by a bagged ensemble of classifiers, and a set of candidate answers gathered from the Internet. Further work includes exploring the inclusion of more information as part of the context, the refinement of the semantic classes for questions and named entities, and the improvement of answer selection methodology.

Acknowledgements. This work was done under partial support of CONACYT (Project Grant U39957-Y), SNI-Mexico, and the Human Language Technologies Laboratory of INAOE. We also like to thanks to the CLEF as well as EFE agency for the resources provided.

References

1. Breiman L. *Bagging predictors. Machine Learning,* 24(2): 123-140, 1996.
2. Burger, J. et al. *Issues, Tasks and Program Structures to Roadmap Research in Question & Answering (Q&A).* NIST 2001.
3. Carreras, X. and Padró, L. *A Flexible Distributed Architecture for Natural Language Analyzers.* In Proceedings of the LREC'02, Las Palmas de Gran Canaria, Spain, 2002.
4. Cowie J., et al., *Automatic Question Answering,* Proceedings of the International Conference on Multimedia Information Retrieval (RIAO 2000)., 2000.
5. Hirshman L. and Gaizauskas R. *Natural Language Question Answering: The View from Here,* Natural Language Engineering 7, 2001.
6. Magnini B., Romagnoli S., Vallin A., Herrera J., Peñas A., Peinado V., Verdejo F. and Rijke M. *The Multiple Language Question Answering Track at CLEF 2003.* CLEF 2003 Workshop, Springer-Verlag.
7. Pérez-Coutiño M., Solorio T., Montes-y-Gómez M., López-López A. and Villaseñor-Pineda L., *Toward a Document Model for Question Answering Systems.* In Advances in Web Intelligence. LNAI3034 Springer-Verlag 2004.
8. Prager J., Radev D., Brown E., Coden A. and Samn V. *The Use of Predictive Annotation for Question Answering in TREC8.* NIST 1999.
9. Quinlan J. R. *C4.5: Programs for machine learning.* 1993. San Mateo, CA: Morgan Kaufmann.

10. Ravichandran D. and Hovy E. *Learning Surface Text Patterns for a Question Answering System.* In ACL Conference, 2002.
11. Solorio T., Pérez-Coutiño M., Montes-y-Gómez M., Villaseñor-Pineda L., and López-Lopez A. 2004. *A language independent method for question classification.* In COLING-04. 2004. Switzerland.
12. Vapnik, V. *The Nature of Statistical Learning Theory,* Springer, 1995.
13. Vicedo, J.L., Izquierdo R., Llopis F. and Muñoz R., *Question Answering in Spanish.* CLEF 2003 Workshop, Springer-Verlag.
14. Vicedo, J.L., Rodríguez, H., Peñas, A. and Massot, M. Los sistemas de Búsqueda de Respuestas desde una perspectiva actual. Revista de la Sociedad Española para el Procesamiento del Lenguaje Natural, n.31, 2003.
15. Witten H. and Frank E. 1999. *Data Mining, Practical Machine Learning Tools and Techniques with Java Implementations.* The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann.

A Max-SAT Solver with Lazy Data Structures

Teresa Alsinet, Felip Manyà, and Jordi Planes

Departament d'Informàtica
Universitat de Lleida,Jaume II, 69, 25001-Lleida, Spain
{tracy, felip, jordi}@eps.udl.es

Abstract. We present a new branch and bound algorithm for Max-SAT which incorporates original lazy data structures, a new variable selection heuristics and a lower bound of better quality. We provide experimental evidence that our solver outperforms some of the best performing Max-SAT solvers on a wide range of instances.

Keywords: Max-SAT, branch and bound, lower bound, heuristics, data structures.

1 Introduction

In recent years we have seen an increasing interest in propositional satisfiability (SAT) that has led to the development of fast and sophisticated complete SAT solvers like Chaff, Grasp, RelSat and Satz, which are based on the well-known Davis-Putnam-Logemann-Loveland (DPLL) procedure [5]. Given a Boolean CNF formula ϕ, such algorithms determine whether there is a truth assignment that satisfies ϕ. Unfortunately, they are not able to solve a well-known satisfiability optimization problems: Max-SAT. Given a Boolean CNF formula ϕ, Max-SAT consists of finding a truth assignment that maximizes the number of satisfied clauses in ϕ. When all the clauses have at most k literals per clause Max-SAT is called Max-k-SAT.

To our best knowledge, there are only three exact algorithms for Max-SAT that are variants of the DPLL procedure. The first was developed by Wallace & Freuder [15] (WF), the second was developed by Borchers & Furman [3] (BF), and the third, which is based on BF, was developed by Alsinet, Manyà & Planes [2] (AMP). All of them are depth-first branch and bound algorithms. The first was implemented in Lisp, while the rest were implemented in C and are publicly available. There are other exact algorithms for Max-SAT, but based on mathematical programming techniques [4, 6, 8]. There are also two exact DPLL-based algorithms for solving Max-2-SAT: one is due to Zhang, Shen & Manyà [16] (ZSM), and the other to Alber, Gramm & Niedermeier [1] (AGN).

In this paper we first present a new branch and bound algorithm for Max-SAT which incorporates original lazy data structures, a new variable selection heuristic, and a lower bound of better quality. We then report on an experimental investigation we have conducted in order to evaluate our solver on Max-SAT instances. The results obtained provide experimental evidence that our solver outperforms some of the best performing existing Max-SAT solvers on a wide range of instances.

Our new Max-SAT solver, which we call Lazy, differs from BF and AMP in the data structures used to represent and manipulate CNF formulas, in the simplification preprocessing techniques applied, in the lower bound, in the variable selection heuristic and in the incorporation of the Dominating Unit Clause (DUC) rule.

2 Branch and Bound for Max-SAT

The space of all possible assignments for a CNF formula ϕ can be represented as a search tree, where internal nodes represent partial assignments and leaf nodes represent complete assignments. A branch and bound algorithm for Max-SAT explores the search tree in a depth-first manner. At every node, the algorithm compares the number of clauses unsatisfied by the best complete assignment found so far —called upper bound *(UB)*— with the number of clauses unsatisfied by the current partial assignment *(unsat)* plus an underestimation of the number of clauses that become unsatisfied if we extend the current partial assignment into a complete assignment *(underestimation)*. The sum *unsat* + *underestimation* is called lower bound *(LB)*. Obviously, if $UB \leq LB$, a better assignment cannot be found from this point in search. In that case, the algorithm prunes the subtree below the current node and backtracks to a higher level in the search tree. If *UB* > *LB*, it extends the current partial assignment by instantiating one more variable; which leads to create two branches from the current branch: the left branch corresponds to instantiate the new variable to false, and the right branch corresponds to instantiate the new variable to true. In that case, the formula associated with the left (right) branch is obtained from the formula of the current node by deleting all the clauses containing the literal $\neg p$ (p) and removing all the occurrences of the literal p $(\neg p)$; i.e., the algorithm applies the one-literal rule [9]. The solution to Max-SAT is the value that *UB* takes after exploring the entire search tree.

Borchers & Furman [3] designed and implemented a branch and bound solver for Max-SAT, called BF herein, that incorporates two quite significant improvements:

- Before starting to explore the search tree, they obtain an upper bound on the number of unsatisfied clauses in an optimal solution using the local search procedure GSAT [13].
- When branching is done, branch and bound algorithms for Max-SAT apply the one-literal rule (simplifying with the branching literal) instead of applying unit propagation as in the DPLL-style solvers for SAT.[1] If unit propagation is applied at each node, the algorithm can return a non-optimal solution. However, when the difference between the lower bound and the up-

[1] By unit propagation we mean the repeated application of the one-literal rule until a saturation state is reached.

per bound is one, unit propagation can be safely applied, because otherwise by fixing to false any literal of any unit clause we reach the upper bound. Borchers & Furman perform unit propagation in that case.

The improvements incorporated into BF are also used in the rest of DPLL-based Max-SAT solvers considered in this paper (i.e. AMP, ZSM, AGN).
The lower bound and variable selection heuristic of BF are:

- $\mathtt{LB}_{\mathrm{BF}} = \mathit{unsat}$. Note that that the number of clauses unsatisfied by the current partial assignment coincides with the number of empty clauses that the formula associated with the current partial assignment contains. In this elementary lower bound there is no underestimation of the number of clauses that become unsatisfied if we extend the current partial assignment into a complete assignment.
- `MOMS` [12]: selects a variable among those that appear more often in clauses of minimum size. That is the heuristic of Borchers&Furman. Ties are broken by choosing the first variable in lexicographical order.

In a previous paper [2], we incorporated two improvements into BF that led to significant performance improvements: a lower bound of better quality ($\mathtt{LB}_{\mathrm{AMP}}$) and another variable selection heuristic (JW):

- $\mathtt{LB}_{\mathrm{AMP}} = \mathit{unsat} + \sum_{p \in \phi'} \mathit{min}(\mathit{ic}(p), \mathit{ic}(\neg p))$,where ϕ' is the formula associated with the current partial assignment, and $\mathit{ic}(p)$ ($\mathit{ic}(\neg p)$) —inconsistency count of p ($\neg p$)— is the number of clauses that become unsatisfied if the current partial assignment is extended by fixing p to true (false). Note that $\mathit{ic}(p)$ ($\mathit{ic}(\neg p)$) coincides with the number of unit clauses of ϕ' that contain $\neg p$ (p).

  If for each variable we count the number of positive and negative literals in unit clauses, we can know the number of unit clauses that will not be satisfied if the variable is instantiated to true or false. Obviously, the total number of unsatisfied clauses resulting from either instantiation of the variable must be greater than or equal to the minimum count. Moreover, the counts for different variables are independent, since they refer to different unit clauses. Hence, by summing the minimum count for all variables in unit clauses and adding this sum to the number of empty clauses, we calculate a lower bound for the number of unsatisfied clauses given the current assignment. Such a lower bound was considered in [15].
- Jeroslow-Wang (`JW`) [7]: given a formula ϕ, for each literal l of ϕ the following function is defined: $J(l) = \sum_{l \in C \in \phi} 2^{-|C|}$, where $|C|$ is the length of clause C. `JW` selects a variable p of ϕ among those that maximize $J(p) + J(\neg p)$.

Our solver AMP is basically BF with the above lower bound and variable selection heuristic. Figure 1 shows the pseudo-code of the skeleton of BF and AMP. We use the following notation: `empty-clauses`(ϕ) is a function that returns the number of empty clauses in ϕ; `lower-bound`(ϕ) is the sum of the number of empty clauses in ϕ plus an underestimation of the number of unsatisfied clauses in the formula obtained from ϕ by removing its empty clauses. In our case, $\mathtt{LB}_{\mathrm{BF}}$ or $\mathtt{LB}_{\mathrm{AMP}}$; ub is an upper bound of the number of unsatisfied clauses

```
Input: max-sat($\phi$, $ub$) : A Boolean CNF formula $\phi$ and an upper bound $ub$
 1: if $\phi = \emptyset$ or $\phi$ only contains empty clauses then
 2:     return empty-clauses($\phi$)
 3: end if
 4: if lower-bound($\phi$) $\geq ub$ then
 5:     return $\infty$
 6: end if
 7: if $unsat = ub - 1$ then
 8:     $\phi \leftarrow$ unit-propagation($\phi$)
 9: end if
10: $p \leftarrow$ select-variable($\phi$)
11: $ub \leftarrow \min(ub,$ max-sat$(\phi_{\neg p}, ub))$
12: return $\min(ub,$ max-sat$(\phi_p, ub))$
Output: The minimum number of clauses of $\phi$ that can be unsatisfied
```

Fig. 1. Branch and Bound for Max-SAT

in an optimal solution. We assume that the input value is that obtained with GSAT; `select-variable`(ϕ) is a function that returns a variable of ϕ following an heuristic; in our case, MOMS or JW; and ϕ_p ($\phi_{\neg p}$) is the formula obtained by applying the one-literal rule to ϕ using the literal p ($\neg p$).

2.1 A New Max-SAT Solver

Our new Max-SAT solver, which we call Lazy, differs from previous solvers in the data structures used to represent and manipulate CNF formulas, in the lower bound, in a novel variable selection heuristic, in the preprocessing of such formulas and in the incorporation of the Dominating Unit Clause (DUC) rule. Lazy was implemented in C++.

Data Structures. BF and AMP use adjacency lists to represent CNF formulas and their variable selection heuristics are dynamic. Lazy uses a static variable selection heuristic (defined below) that allows us to implement extremely efficient data structures for representing and manipulating CNF formulas. Our data structures take into account the following fact: we are only interested in knowing when a clause has become unit or empty. Thus, if we have a clause with four variables, we do not perform any operation in that clause until three of the variables appearing in the clause have been instantiated; i.e., we delay the evaluation of a clause with k variables until $k - 1$ variables have been instantiated. In our case, as we instantiate the variables using a static order, we do not have to evaluate a clause until the penultimate variable of the clause in the static order has been instantiated.

The data structures are defined as follows: For each clause we have a pointer to the penultimate variable of the clause in the static order, and the clauses of a CNF formula are ordered by that pointer. We also have a pointer to the last variable of the clause. When a variable p is fixed to true (false), only the clauses whose penultimate variable in the static order is $\neg p$ (p) are evaluated. This approach has two advantages: the cost of backtracking is constant (we do

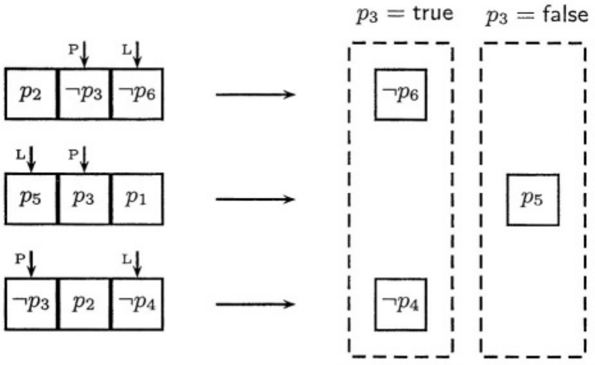


Fig. 2. Lazy data structure snapshot example. The arrow L stands for the pointer to the last variable and the arrow P stands for the pointer to the penultimate variable

not have to undo pointers like in adjacency lists) and, at each step, we evaluate a minimum number of clauses.

For instance, suppose we have a formula with the following clauses and that variables are instantiated in lexicographic order:

$$p_2 \vee \neg p_3 \vee \neg p_6$$
$$p_5 \vee p_3 \vee p_1$$
$$\neg p_3 \vee p_2 \vee \neg p_4$$

If p_1 and p_2 have been instantiated to false, when we branch on $p_3 = true$ we derive the unit clauses $\neg p_6$ and $\neg p_4$; and when we branch on $p_3 = false$ we derive the unit clause p_5. The data structure snapshot is shown in Fig. 2.

Lower Bound. Lazy incorporates a lower bound ($\mathrm{LB_{LAZY}}$) of better quality than $\mathrm{LB_{BF}}$ and $\mathrm{LB_{AMP}}$. $\mathrm{LB_{LAZY}}$ can be understood as $\mathrm{LB_{AMP}}$ extended with a specialization of the so-called star rule defined in [11]. The star rule states that if we have a clause of the form $l_1 \vee \cdots \vee l_k$, where $l_1, \ldots, l_k$ are literals, and k unit clauses of the form $\neg l_1, \ldots, \neg l_k$, then the lower bound can be incremented by one. In our case, we only consider clauses of length two. For longer clauses the star rule did not lead to performance improvements in our experimental investigation. The pseudo-code of $\mathrm{LB_{LAZY}}$ is defined as follows [14]:

```
1: $\mathrm{LB_{LAZY}} := \mathrm{LB_{AMP}}$
2: for every clause $l_1 \vee l_2 \in \phi$ do
3:    if $\neg l_1 \in \phi$ and $\neg l_2 \in \phi$ then
4:       $\mathrm{LB_{LAZY}} := \mathrm{LB_{LAZY}} + 1$
5:       $\phi := \phi - \{l_1 \vee l_2, \neg l_1, \neg l_2\}$
6:    end if
7: end for
```

Variable Selection Heuristic. MOMS is a branching heuristic that selects a variable among those that appear more often in clauses of minimum size, and breaks ties by choosing the first variable in lexicographical order. We have defined MOMS*, which is like MOMS but breaks ties in a different way. In MOMS* ties are broken by choosing the variable p with the highest weight w. Such a weight is defined as follows: $w(p) = \prod_{l \in S_p} \text{occur}(l)$, where S_p denotes the set of negated neighbouring literals of variable p, and $\text{occur}(l)$ denotes the number of occurrences of literal l. A literal $\neg l$ is a negated neighbouring literal of variable p if l occurs in a clause that contains the literal p or the literal $\neg p$. The reason behind that calculation is that any clause $l_1 \vee l_2 \vee l_3$ can be seen as the implication $\neg l_1 \wedge \neg l_2 \rightarrow l_3$. So, the greater the product $\text{occur}(\neg l_1) \cdot \text{occur}(\neg l_2)$, the higher the probability of creating unit clause l_3.

Note that MOMS is used as a dynamic variable selection heuristic in BF while MOMS* is used as a static variable selection heuristic in Lazy.

Formula Reduction Preprocessing. Before the search starts, the initial formula is simplified by applying the resolution rule to some binary clauses. For every pair of clauses $p_1 \vee p_2$ and $\neg p_1 \vee p_2$ such that variable p_1 precedes variable p_2 in the static instantiation order, Lazy reduces them to the unit clause p_2, and for every pair of clauses $p_1 \vee \neg p_2$ and $\neg p_1 \vee \neg p_2$ such that variable p_1 precedes variable p_2 in the static instantiation order, Lazy reduces them to the unit clause $\neg p_2$.

Dominating Unit Clause (DUC) Rule. DUC is an inference rule, defined in [11], that allows us to fix the truth value of a variable; i.e., it avoids to apply branching on that variable. DUC is defined as follows: If a CNF formula ϕ has k occurrences of a literal p ($\neg p$) and has at least k unit clauses of the form $\neg p$ (p), then the value of p can be set to false (true).

3 Experimental Results

We conducted an experimental investigation in order to compare the performance of BF, AMP, and Lazy. When dealing with Max-2-SAT instances, we also compare with ZSM and AGN. The experiments were performed on a 2GHz Pentium IV with 512 Mb of RAM under Linux.

In our first experiment, we evaluated the relevance of defining lazy data structures to get substantial performance improvements. To this end, we compared BF and Lazy using a simple variable selection heuristic: variables are instantiated in lexicographical order. Moreover, we removed all the improvements we introduced into Lazy and replaced LB_{LAZY} with LB_{BF}. In this way, we have that BF and Lazy traverse the same search tree. Figure 3 shows the results obtained when solving sets of randomly generated Max-3-SAT instances with 30 variables and a different number of clauses. Such instances were generated using the method described in [10]. We generated sets for 180, 240, 300, 360, 420 and 480 clauses, where each set had 100 instances. We observe that this modified version of Lazy is about 5 times faster than BF when both solvers traverse the same search tree.

In our second experiment, we generated sets of random Max-2-SAT instances with 50 and 100 variables and a different number of clauses. Such instances were

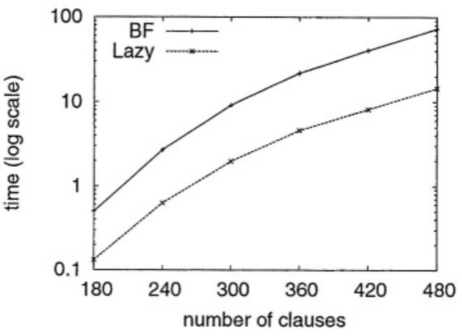


Fig. 3. Experimental results for 30-variable Max-3-SAT instances. Mean time (in seconds)

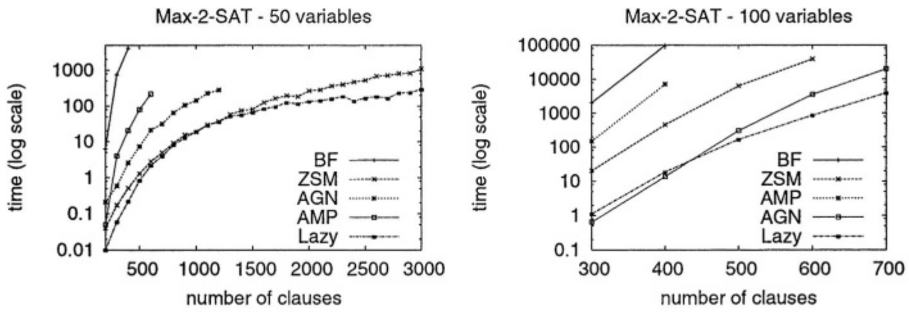


Fig. 4. Experimental results for 50-variable and 100-variable Max-2-SAT instances. Mean time (in seconds)

generated using the method described in [10], and each set had 100 instances. The results of solving such instances with BF, AMP, ZSM, AGN and Lazy are shown in Fig. 4. Along the horizontal axis is the number of clauses, and along the vertical axis is the mean and median time (in seconds) needed to solve an instance of a set. Notice that we use a log scale to represent run-time. Clearly, Lazy outperforms the rest of solvers, even ZSM and AGN that are specifically designed to solve Max-2-SAT instances.

In our third experiment, we generated sets of random Max-3-SAT and Max-4-SAT instances with 50 variables and a different number of clauses. Such instances were generated using the method described in [10], and each set had 100 instances. The results of solving such instances with BF, AMP, and Lazy are shown in Fig. 5. Again, we clearly see that Lazy provides substantial performance improvements.

The performance improvements of Lazy are due to its lazy data structures, the simplification preprocessing techniques applied, the quality of the lower bound, the incorporation of the dominating unit clause rule, and the variable

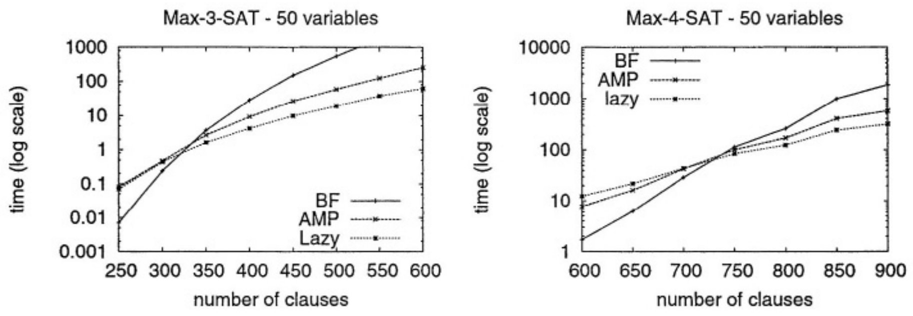


Fig. 5. Experimental results for 50-variable Max-3-SAT (left) and Max-4-SAT (right) instances. Mean time (in seconds)

selection heuristic used. We believe that our results could be further improved by adapting the lazy data structures defined in the paper to deal with dynamic variable selection heuristics. It would also be interesting to test Lazy on more realistic benchmarks.

Acknowledgments. Research partially supported by projects TIC2003-00950 and TIC2001-1577-C03-03 funded by the *Ministerio de Ciencia y Tecnología.*

References

1. J. Alber, J. Gramm, and R. Niedermeier. Faster exact algorithms for hard problems: A parameterized point of view. In *25th Conf. on Current Trends in Theory and Practice of Informatics,* LNCS, pages 168–185. Springer-Verlag, November 1998.
2. T. Alsinet, F. Manyà, and J. Planes. Improved Branch and Bound Algorithms for Max-SAT. In *Proc. of Sixth Int. Conf. on the Theory and Applications of Satisfiability Testing,* May 2003.
3. B. Borchers and J. Furman. A two-phase exact algorithm for MAX-SAT and weighted MAX-SAT problems. *Journal of Combinatorial Optimization,* 2:299–306, 1999.
4. J. Cheriyan, W. Cunningham, L. Tunçel, and Y. Wang. A linear programming and rounding approach to MAX-2-SAT. In D. Johnson and M. Trick, editors, *Cliques, Coloring and Satisfiability,* volume 26, pages 395–414. DIMACS, 1996.
5. M. Davis, G. Logemann, and D. Loveland. A machine program for theorem-proving. *Communications of the ACM,* 5:394–397, 1962.
6. E. de Klerk and J. P. Warners. Semidefinite programming approaches for MAX-2-SAT and MAX-3-SAT: computational perspectives. Technical report, Delft, The Netherlands, 1998.
7. R. G. Jeroslow and J. Wang. Solving propositional satisfiability problems. *Annals of Mathematics and Artificial Intelligence,* 1:167–187, 1990.
8. S. Joy, J. Mitchell, and B. Borchers. A branch and cut algorithm for MAX-SAT and Weighted MAX-SAT. In *DIMACS Workshop on Satisfiability: Theory and Applications,* 1996.

9. D. W. Loveland. *Automated Theorem Proving. A Logical Basis,* volume 6 of *Fundamental Studies in Computer Science.* North-Holland, 1978.
10. D. Mitchell, B. Selman, and H. Levesque. Hard and easy distributions of SAT problems. In *Proceedings of the 10th National Conference on Artificial Intelligence, AAAI'92, San Jose/CA, USA,* pages 459–465. AAAI Press, 1992.
11. R. Niedermeier and P. Rossmanith. New upper bounds for maximum satisfiability. *Journal of Algorithms,* 36:63–88, 2000.
12. D. Pretolani. Efficiency and stability of hypergraph SAT algorithms. In *Proceedings of the DIMACS Challenge II Workshop,* 1993.
13. B. Selman, H. Levesque, and D. Mitchell. A new method for solving hard satisfiability problems. In *Proceedings of the 10th National Conference on Artificial Intelligence, AAAI'92, San Jose/CA, USA,* pages 440–446. AAAI Press, 1992.
14. H. Shen and H. Zhang. Improving exact algorithms for MAX-2-SAT. In *Proceedings of the 8th International Symposium on Artificial Intelligence and Mathematics,* 2004.
15. R. Wallace and E. Freuder. Comparative studies of constraint satisfaction and Davis-Putnam algorithms for maximum satisfiability problems. In D. Johnson and M. Trick, editors, *Cliques, Coloring and Satisfiability,* volume 26, pages 587–615. DIMACS, 1996.
16. H. Zhang, H. Shen, and F. Manya. Exact algorithms for MAX-SAT. In *4th Int. Workshop on First order Theorem Proving,* June 2003.

Three Valued Logic of Łukasiewicz for Modeling Semantics of Logic Programs

Mauricio Osorio[1], Verónica Borja[2], and José Arrazola[2]

[1] Universidad de las Américas,
CENTIA Sta. Catarina Mártir, Cholula,
Puebla 72820 México
[2] Benemérita Universidad Autónoma de Puebla,
FCFM Blvd. Sn. Claudio y 18 Sur, Puebla,
Puebla 72560 México
josorio@mail.udlap.mx, vero0304@hotmail.com, arrazola@fcfm.buap.mx

Abstract. In order to really understand all aspects of logic-based program development of different semantics, it would be useful to have a common solid logical foundation.

The stable semantics are based on G_3 but we show that stable semantics can be fully represented in the three valued logic of Łukasiewicz. We construct a particular semantics that we call Ł3-WFS wich is defined over general propositional theories, can be defined via three valued logic of Łukasiewicz. Interesting Ł3-WFS seems to satisfy most of the principles of a well behaved semantics. Hence we propose the three valued Łukasiewicz logic to model WFS, extensions of WFS, and the Stable semantics.

1 Introduction

A-Prolog (Stable Logic Programming [11] or Answer Set Programming) is the realization of much theoretical work on Nonmonotonic Reasoning and AI applications of Logic Programming (LP) in the last 15 years. This is an important logic programming paradigm that has now broad acceptance in the community. Efficient software to compute answer sets and different applications to model real life problems justify this assertion.

The well founded semantics is a very well known paradigm originated at the same time that stable semantics [20]. The main difference between STABLE and WFS is in the definition of the former, a *guess* is made and then a particular (2-valued) model is constructed and used to justify the guess or to reject it. However, in the definition of WFS, more and more atoms are declared to be true (or false): once a decision has been drawn, it will never be rejected. WFS is based on a single 3-valued intended model.

Several authors have recognized the interest in semantics with closed behavior to classical logic see [5–8, 19]. They have extend the WFS semantics by putting an additional mechanism on top of its definition. Dix noticed that the new semantics sometimes have a more serious shortcomings than WFS and hence he defined a set of principles where all semantics should be checked against [7]. It is worth to mention that such notions helped Dix to propose the concept of well behaved semantics. We think that is important to understand well such concept if one wants to follow any serious methodology for logic-based program development.

We introduce an extension of WFS that we will call Ł3-WFS with the following properties:

1. It is defined based on completions (as the stable semantics) but using the well known three valued logic of Łukasiewicz.
2. Ł3-WFS is defined for propositional theories based in basic formulas.
3. Using the knowledge ordering ($\leq_k$, see [6]), we have that WFS $<$ Ł3-WFS and also WFS$^+$ $<$ Ł3-WFS defined by Dix also satisfies this property.
4. The known counter examples for the well behavior of several known extensions of WFS (such as GWFS and EWFS) do not apply for Ł3-WFS. We conjeture that Ł3-WFS satisfies several of the principles given for well behaved semantics [2].
5. Ł3-WFS is different from GWFS, EWFS, WFS$^+$.

We expect the reader to have some familiarity with many valued logics of Łukasiewicz and logic programming.

2 Background

We consider a formal (propositional) language built from an alphabet containing: a denumerable set $\mathcal{L}$ of elements called atoms, the standard 2-place connectives $\wedge$, $\vee$, $\rightarrow$, and the 1-place connective $\neg$. Formulas and theories are constructed as usual in logic. In this paper we only consider finite theories. We will later define other connectives, but only for temporal use.

We define the class of *basic formulas*[1] recursively as follows:

$$\begin{array}{ll} \neg a,\ a & \text{if } a \text{ is an atom.} \\ \alpha \vee \beta & \text{if } \alpha, \beta \text{ are basic formulas.} \\ \alpha \wedge \beta & \text{if } \alpha, \beta \text{ are basic formulas.} \\ \alpha \rightarrow \beta & \text{if } \alpha, \beta \text{ are basic formulas.} \end{array}$$

A *normal program* is a set of rules of the form

$$A_1 \wedge ... \wedge A_m \wedge \neg A_{m+1} \wedge ... \wedge \neg A_n \rightarrow A_0$$

where each A_i for $i = \overline{0, n}$ is an atom.

We use the well known definition of a stratified program, see [1]. We use the notation $\vdash_X F$ to denote that the formula F is provable (a theorem or tautology) in logic X. If T is a theory we use the symbol $T \vdash_X F$ to denote $\vdash_X (F_1 \wedge \cdots \wedge F_n) \rightarrow F$ for some formulas $F_i \in T$. We say that a theory T is *consistent* if $T \not\vdash_X \bot$. We also introduce, if T and U are two theories, the symbol

[1] this class of formulas will be used in sections 4-6.

$T \vdash_X U$ to denote that $T \vdash_X F$ for all formulas $F \in U$. We will write $T \Vdash_X U$ to denote the fact that (i) T is consistent and (ii) $T \vdash_X U$.

Given a class of programs C, a *semantic operator* Sem is a function that assigns to each program $P \in C$ a set of sets of atoms $M \subseteq \mathcal{L}_P$. These sets of atoms are usually some "preferred" two valued models of the program P each of them is called a *Sem model* of P. Sometimes, we say, that this is the scenarios semantics [8]. Given a scenarios semantics Sem, we define the scepticalsemantics of a program P as: $\mathrm{Sem}(P) = \bigcap\{M \cup \neg \widetilde{of}\ P\}$, where $\widetilde{M} = \mathcal{L}_P \setminus M$ and $\neg M = \{\neg a : a \in M\}$. Given two scenarios semantics S_1 and S_2, wedefine: $S_1 \leq S_2$ if for every program P is true that $S_1(P) \subseteq S_2(P)$. We can easily define $S_1 = S_2$ and $S_1 < S_2$. We say that a semantics is stronger than another one according to this order.

3 Three Valued Logic of Łukasiewicz

The Polish logician and philosopher Jan Łukasiewicz began to create systems ofmany-valued logic in 1920, particularly a system with a third value for "possible" and to model in this way the modalities "it is necessary that" and "it is possible that"[2]. The outcome of these investigations arethe Łukasiewicz systems, and a series of theoretical results concerning these systems [3].

To construct Ł$_3$ we consider a formal (propositional) language built from an alphabet containing: a denumerable set $\mathcal{L}$ of elements called atoms, the 2-place connective $\rightarrow_{Ł}$, and the 1-place connective $\neg_{Ł}$. The logical constants $\bot$, $\top$, the connectives $\vee_{Ł}$, $\wedge_{Ł}$, and the modal operators $\Diamond_{Ł}$ and $\Box_{Ł}$ are defined as follows:

[2] Łukasiewicz tried to deal with Aristotle's paradox of the seabattle: "Two admirals, A and B, are preparing their navies for a sea battletomorrow. The battle will be fought until one side is victorious. But the'laws' of the excluded middle (no third truth-value) and of noncontradiction(not both truth-values), mandate that one of the propositions, 'A wins' and'B wins', is true (always has been and ever will be) and the other is false(always has been and ever will be). Suppose 'A wins' is today true. Thenwhatever A does (or fails to do) today will make no difference; similarly,whatever B does (or fails to do) today will make no difference: the outcomeis already settled. Or again, suppose 'A wins' is today false. Then nomatter what A does today (or fails to do), it will make no difference;similarly, no matter what B does (or fails to do), it will make no difference: the outcome is already settled. Thus, if propositions bear their truth-values timelessly (or unchangingly and eternally), then planning, or as Aristotle put it 'taking care', is illusory in its efficacy. The future will be what it will be, irrespective of our planning, intentions, etc." (For more references about this paradox see: Taylor, Richard (1957). *"The problem of future contingencies."* Philosophical Review 66: 1-28, or visit *http://www2.msstate.edu/ ~ jjs87/SO/1093/freedom.html*)

[3] *http://en.wikipedia.org/wiki/Multi – valued_logic*

$$\begin{array}{llll} \top & := (p \rightarrow_{Ł} p) & \alpha \wedge_{Ł} \beta & := \neg_{Ł}(\neg_{Ł}\alpha \vee_{Ł} \neg_{Ł}\beta) \\ \bot & := \neg_{Ł}\top & \Diamond_{Ł}\alpha & := \neg_{Ł}\alpha \rightarrow_{Ł} \alpha \\ \neg_{Ł}\alpha & := \alpha \rightarrow_{Ł} \bot & \Box_{Ł}\alpha & := \neg_{Ł}(\alpha \rightarrow_{Ł} \neg_{Ł}\alpha) \\ \alpha \vee_{Ł} \beta & := (\alpha \rightarrow_{Ł} \beta) \rightarrow_{Ł} \beta & & \end{array}$$

Formulas and theories are constructed as usual in logic.

A tri-valued valuation v for $\mathcal{L}$ is a map $v : \mathcal{L} \rightarrow \{0, 1, 2\}$ inductively extended over the set of formulas into $\{0, 1, 2\}$ as follows:

- $v(\bot) = 0$
- $v(\neg_{Ł}\alpha) = 2 - v(\alpha)$
- $v(\alpha \rightarrow_{Ł} \beta) = min\{2, 2 - v(\alpha) + v(\beta)\}$

Then in Ł3 for the valuation defined tautologies will be the formulas whose truth value is 2.

In [14] a syntactic characterization of the modal content of Ł3 is studied and the behavior of modal operators are checked against some of the relevant modal principles. Minari also studies Ł3's axiomatization is as well as its relation with modal logics, particularly with S5. An axiomatization for Ł3 over $\mathcal{L}$ is given by the axiom schemes:

- (L_1) $\alpha \rightarrow_{Ł} (\beta \rightarrow_{Ł} \alpha)$
- (L_2) $(\alpha \rightarrow_{Ł} \beta) \rightarrow_{Ł} ((\beta \rightarrow_{Ł} \gamma) \rightarrow_{Ł} (\alpha \rightarrow_{Ł} \gamma))$
- (L_3) $(\neg_{Ł}\beta \rightarrow_{Ł} \neg_{Ł}\alpha) \rightarrow_{Ł} (\alpha \rightarrow_{Ł} \beta)$
- (L_4) $((\alpha \rightarrow_{Ł} \neg_{Ł}\alpha) \rightarrow_{Ł} \alpha) \rightarrow_{Ł} \alpha)$

and the inference rule *Modus ponens*

- (MP) If α, $\alpha \rightarrow_{Ł} \beta \in$ Ł3 then $\beta \in$ Ł3

4 Definition of ASP-WFS Via Ł3

We first explain how are we going to use Ł3 to define our semantics. Stable semantics is given in terms of $\neg_{G_3}$ and $\rightarrow_{G_3}$[4], but in order to define our Ł3-WFS semantics we propose an extra connective $\neg_{G'_3}$. It is important to note that these connectives are abbreviation forms using the standard language of Ł3.

In table 4 you can see the correspondence between the abbreviations and the valuation for the connective $(\neg_{G'_3})$. The reader can easily check the correspondence of truth values of $a \rightarrow_{G_3} b$ and its abbreviation form in the language of Ł3.

Then we have two different 'negations' and then two different logics, namely the original G_3 (with its standard connectives $\neg_{G_3}$ and $\rightarrow_{G_3}$) and G'_3 (with the connectives $\neg_{G'_3}$ and $\rightarrow_{G_3}$).

Definition 1. *Let P be a logical program and M be a set of atoms. M is a G'_3-stable model of P if $\bigwedge_{G'_3}(P \cup \neg_{G'_3}\widetilde{M})$ is consistent and $P \cup \neg_{G'_3}\widetilde{M} \models_{G'_3} M$. (in the sense that $\models_{G'_3} \bigwedge_{G'_3}(P \cup \neg_{G'_3}\widetilde{M}) \rightarrow_{G_3} \bigwedge_{G'_3} M$).*

[4] Note that $\wedge_{G_3}$ corresponds to $\wedge_{Ł}$ and $\vee_{G_3}$ corresponds to $\vee_{Ł}$.

Table 1. Abbreviation Forms in Ł3

$$\begin{array}{rcl}
\neg_{G_3} a & := & \Box_Ł \neg_Ł a \\
a \rightarrow_{G_3} b & := & (a \rightarrow_Ł b) \wedge_Ł \neg_{G_3} \neg_{G_3} (\neg_{G_3} \neg_{G_3} a \rightarrow_Ł b) \\
\neg_{G'_3} a & := & \neg_Ł \Box_Ł a
\end{array}$$

Table 2. Valuation of Connectives $\neg_{G_3}$ and $\neg_{G'_3}$

a	$\neg_Ł a$	$\Box_Ł \neg_Ł a$	$\neg_{G_3} a$
0	2	2	2
1	1	0	0
2	0	0	0

a	$\Box_Ł a$	$\neg_Ł \Box_Ł a$	$\neg_{G'_3} a$
0	0	2	2
1	0	2	2
2	2	0	0

Definition 2. *Let P be a logical program based on basic formulas and M be a set of atoms. M is a Ł3-WFS model of P if M is a G'_3-stable model of P.*

The following is a well known result (see [16, 17, 18]), which characterize stable models for propositional theories in terms of G_3:

Theorem 1. *Let P be a logical program and M be a set of atoms. M is a stable model of P iff $P \cup \neg_{G_3}\widetilde{M} \cup \neg_{G_3}\neg_{G_3} M \Vdash_{G_3} M$.*

But if we reduce the class of programs to disjunctive programs the definition becomes (see [18]):

Theorem 2. *Let P be a disjunctive program and M be a set of atoms. M is a stable model of P iff $P \cup \neg_{G_3}\widetilde{M} \Vdash_{G_3} M$.*

Which is analogous to the previous definition using G'_3. Since G_3 and G'_3 can be expressed in terms of Ł3 using the abbreviations in table 4 then we have two different semantics: stable semantics and Ł3-WFS respectively.

5 Results

We first show that Ł3-WFS is different to some well known semantics and then present its characterization using Ł3 and we finish this section presenting a brief comment on well behaved semantics.

5.1 Comparing Ł3-WFS with Other Semantics

Consider the EWFS semantics, the CUT rule and the following example all of them taken from [6]:

$$\neg a \rightarrow a, \neg x \wedge a \rightarrow b, \neg b \rightarrow y, \neg y \rightarrow z$$

Here EWFS(P) $= \{a, b, \neg x\}$, however EWFS($P \cup \{b\}$) $= \{a, b, z, \neg x, \neg y\}$. This example shows that EWFS does not satisfies CUT. Ł3-WFS(P) $=$

$\{a, b, z, \neg x, \neg y\}$ as well as Ł$_3$-WFS$(P \cup \{b\}) = \{a, b, z, \neg x, \neg y\}$. Hence Ł$_3$-WFS is different to EWFS.

Consider the following two program examples taken from [6]:

$$\neg b \rightarrow p, c \rightarrow b, (p \wedge \neg a) \rightarrow c, \neg b \rightarrow a$$

and

$$\neg b \rightarrow p, (p \wedge \neg a) \rightarrow b, \neg b \rightarrow a$$

One may expect the same semantics of both programs w.r.t. the common language. However, GWFS infers p in the first program, but it does not in the second program. Ł$_3$-WFS gives the same answer in both programs which consists in deriving only $\neg c$ in both programs. Hence, Ł$_3$-WFS is different to GWFS.

Consider the following program example taken from [8]:

$$\neg b \rightarrow a, \neg a \rightarrow b, \neg a \rightarrow x, \neg b \rightarrow x$$

Note that WFS$^+(P) = \{\}$, but AS-WFS$(P) = \{x\}$.

In [8] we have that WFS$^+$ is an stronger extension of WFS, then comparing the semantics we proposed and the results obtained in [6], we have that WFS $<$ Ł$_3$-WFS$<$ STABLE. This can be formalized as follows:

Lemma 1. *Let P be a normal program, then WFS(P) < Ł$_3$-WFS(P) < STABLE(P).*

5.2 Well Behaved Semantics

We conjecture that Ł$_3$-WFS satisfies all principles involved in the definition of a well behaved semantics as long as we reject to interpret $P \cup M$ as P^M. Note that the notion P^M is a syntactic transformation, not required when $P \cup M$ has a logical meaning. Take for instance, the following program P from [7]: $b \rightarrow a, \neg a \rightarrow b$. This example is used to show that WFS$^+$ does not satisfies the Extended Cut principle. While WFS$^+(P) = \{a, \neg b\}$, WFS$^+(P \cup \{\neg b\}) = \{\neg a, \neg b\}$. Moreover, $\{\neg a, \neg b\}$ is neither a 2-valued model, nor a 3-valued model of the program $P \cup \{\neg b\}$. However, this happens because WFS$^+$ does not have a "logical" definition for the semantics of programs extended with constraints (negated formulas). Hence, Dix interprets $P \cup \{\neg b\}$ as $P^{\{\neg b\}}$. $P^{\{\neg b\}} := b \rightarrow a$. Now $\{\neg a, \neg b\}$ is a model of $P \cup \{\neg b\}$. Ł$_3$-WFS has a definition for semantics of any basic propositional theory that allows the use of constraints. In this example we get Ł$_3$-WFS(P) = ASP-WFS$(P \cup \{\neg b\}) = \{a, \neg b\}$. Hence, we propose to reconsider Dix's work on well-behaved semantics, towards a direction of making it more general and logical based.

6 Related Work (Other Modal and Many Valued Logic Characterizations)

In [15] we presented some results about characterizations of stable models and extensions of WFS, such characterizations in terms of the modal S4 and the four-valued bilattice FOUR as well as Gelfond characterization of stable models in S5 are analogous to the characterization in G_3'.

6.1 Via S5

Consider modal logic S5 with its standard connectives that we will denote as: $\sim$, $\rightarrow$, $\vee$ and $\wedge$. McDermott and Doyle introduced a non-monotonic version of S5. They define the X-expansions E of a theory T as those sets satisfying the equation:

$$E = C_{n_X}(T \cup \{\sim \Box\psi \notin E\} \tag{1}$$

where C_{n_X} is the inference operation of the modal logic X. Depending on the approach, an arbitrary selected X-expansion for T or the intersection of all X-expansions for T is considered as a set of nonmonotonic consequences of T. McDermott proved that S5 coincides with its non-monotonic version, hence it is not very interesting. However, he considered formulas to complete the theory. If we only consider adding simple formulas of the form $\sim \Box a$ (a an atom), the story changes. In fact, Gelfond [10] was able to characterize stable models of stratified normal programs using this idea and the following translation: A normal clause:

$$A_1 \wedge ... \wedge A_m \wedge \neg A_{m+1} \wedge ... \wedge \neg A_n \rightarrow A_0 \tag{2}$$

becomes

$$A_1 \wedge ... \wedge A_m \wedge \sim \Box A_{m+1} \wedge ... \wedge \sim \Box A_n \rightarrow A_0 \tag{3}$$

6.2 Via S4

Now consider modal logic S4 with its standard connectives. Let $\neg\alpha$ be the abbreviation form of the modal formula $\sim \Box\alpha$. Gelfond in [10] gives a definition of similar semantics to AS-WFS, but it covers only the class of stratified programs, we generalized this concept in the following definition:

Definition 3. *Let P be a theory based on basic formula and M be a set of atoms. We define M to be an AS-WFS model of P iff* $P \cup \neg\widetilde{M} \Vdash_{S4} M$. *We denote the sceptical semantics of P as AS-WFS(P).*

Hence, this definition opens the research line of defining other WFS extensions via different modal logics. For example, modal logic K behaves 'closer' (but still different) to the stable semantics. Consider the following example: $\neg a \rightarrow b, \neg b \rightarrow a, \neg p \rightarrow a, \neg p \rightarrow p$. Then AS-WFS has two models, namely $\{a, p\}, \{b, p\}$. But using modal logic K we obtain no models.

Lemma 2. *Let P be a theory based on basic formula and M be a set of atoms. Then M is an AS-WFS model of P iff* $P \cup \neg\widetilde{M} \Vdash_{S4} M$.

Is well known that STABLE and WFS agree in the class of normal stratified programs. Hence, we have the following result.

Corollary 1. *If P is a stratified normal program, WFS(P)= AS-WFS(P)= STABLE(P).*

Table 3. FOUR$_{S5}$-valuation

A	$\sim A$
0	3
1	2
2	1
3	0

$\rightarrow$	0	1	2	3
0	3	3	3	3
1	2	3	2	3
2	1	1	3	3
3	0	1	2	3

A	$\Box A$
0	0
1	0
2	0
3	3

Table 4. Abbreviation forms in FOUR

$\sim \alpha$	$:=$	$\sim_{kn}\sim_{tr} \alpha$
$\alpha \rightarrow \beta$	$:=$	$\sim_{kn}\sim_{tr} \alpha \vee_{kn} \beta$
$\Box\alpha$	$:=$	$\alpha\wedge_{kn} \sim_{tr} \alpha$
$\bot$	$:=$	$\sim_{kn}\sim_{tr} a \wedge_{kn} a$ for a given atom a.
$\alpha \vee \beta$	$:=$	$\alpha \vee_{kn} \beta$
$\alpha \wedge \beta$	$:=$	$\alpha \wedge_{kn} \beta$

6.3 Via FOUR

The logical role that the four-valued structure has among Ginsberg's well known bilattices is similar to the role that the two-valued algebras has among Boolean algebras. Four valued semantics is a very suitable setting for computerized reasoning according to Belnap and in fact the original motivation of Ginsberg for introducing bilattices was to provide a uniform approach for a diversity of applications in AI. Bilattices were further investigated by Fitting, who showed that they are useful also for providing semantics to logic programs, hence our interest is focus on relating FOUR with ASP-WFS.

The FOUR-Valuation Bilattice. Belnap introduced a logic for dealing in an useful way with inconsistent and incomplete information. This logic is based on a structure called FOUR, see [3]. This structure has four truth values, the classical t and f, and two new $\top$ that intuitively denotes lack of information (no knowledge), and $\bot$ that indicates inconsistency ("over"-knowledge). These values have two different natural orderings. Measuring the truth: The minimal element is f, the maximal element is t and values $\top$ and $\bot$ are incomparable. Reflecting differences in the amount of knowledge or information: The minimal element is $\bot$, the maximal element is $\top$ and values f and t are incomparable. We read the bilattice FOUR identifying $\bot$ as 0, $\top$ as 3, f as 1 and t as 2. We define the valuation of operators $\sim$, $\rightarrow$ and $\Box$ as in table 4. It is important to note that these connectives are abbreviation forms using the standard language of FOUR as it is shown in table 6.3.

As before, we define our main negation operator (-): $\neg p$ *as* $\sim \Box p$.

Using the reading of FOUR and the valuation defined in the table 4 tautologies will be the formulas whose truth value is 3. Examples of some tautologies are: $(\neg a \rightarrow a) \rightarrow a$, $a \vee \neg a$, $\neg\neg a \rightarrow a$, $a \rightarrow a)$. Note that $a \rightarrow \neg\neg a$ is not a tautology.

Theorem 3. *Let P be a theory based on basic formula and M be a set of atoms. M is an AS- WFS model of P iff* $P \cup \neg\widetilde{M} \Vdash_{\mathrm{FOUR}} M$.

6.4 Models in Ł$_3$ and FOUR

In order to compare Ł$_3$-WFS and AS-WFS semantics it is necessary to study models in terms of G'_3 and $FOUR_{S5}$. Here we have some preliminary results:

Theorem 4. *Let P be a theory based on basic formula and M be a set of atoms. M is an* G'_3*-WFS model of P then exists* M' *an* $FOUR_{S5}$ *model of P.*

The proof is by induction over the length of formula.

Corollary 2. *If* α *is a tautology in* $FOUR_{S5}$ *then* α *is a tautology in* G'_3.

7 Conclusions

There is still actual interest in extensions of WFS ([5, 8]). We need however to find a logical framework to define such extensions if one really believes in a logic-based program development approach. We propose an approach to define extensions of the WFS semantics based on completions with the same spirit of STABLE, hence closing the gap between both approaches. As a result, we gain a better understanding of those semantics as well as the relation among them.

We have that Ł$_3$-WFS is sound with respect to stable models semantics and it can be used to approximate stable entailment. Still it is left work to do with respect to this semantics and our future work is to continue going deep in this semantics to see what properties of the well-behaved semantics it satisfies.

References

1. K. Apt, H. A. Blair and A Walker, Towards a Theory of declarative Knowledge, in J.Minker (ed).*Fundations of deductive data bases.* Morgan Kaufmann, 89 148. 1988.
2. Gerard Brewka, Jürgen Dix and Kurt Konolige. *Non Monotonic Reasoning An Overview.* Center for the Study of Languages and Information Stanford California, 1997.
3. Avron Arnon. *On the Expressive Power of the Three-Valued and FOUR-Valued Languages.* Journal of Logic and Computation 9. 1999.
4. Avron Arnon. *Natural 3-valued Logics-Characterization and Proof Theory* Journal of Symbolic Logic 56, 276-294 (1991).
5. Marc Denecker, Nikolay Pelov, and Maurice Bruynooghe. *Ultimate Well-Founded and Stable Semantics for Logic Programs with Aggregates.* Logic programming. *Proceedings of the 17th International Conference, ICLP 2001,* pages 212-226 LNCS 2237, Springer, November/December 2001.
6. Jürgen. Dix. A Classification Theory of Semantics of Normal Logic Programs: I. Strong Properties. Fundamental Informaticae XXII (3)* 227-255, 1995.

7. Jürgen Dix. A Classification Theory of Semantics of Normal Logic Programs: II. Weak Properties. Fundamental Informaticae XXII (3)* 257-288, 1995.
8. Jürgen Dix, Mauricio Osorio, and Claudia Zepeda. A general theory of confluent rewriting systems for logic programming and its applications. *Annals of Pure and Applied Logic,* 108(1–3):153–188, 2001.
9. Josep Maria Font, Petr Hajek. *Łukasiewicz and modal logic.* Preprints.
10. Michael Gelfond. *On stratified auto-epistemic theories.* In Proceedings of AAAI-87.pp 207-211. Morgan Kaufmann, 1987.
11. Michael Gelfond, Vladimir Lifschitz. The stable model semantics for logic programs. Proceedings of the Fifth International Conference on Logic Programming MIT Press. Cambridge, Ma. 1988. pp.1070-1080.
12. Vladimir Lifschitz, David Pearce, and Agustín Valverde. Strongly equivalent logic programs. ACM Transactions on Computational Logic, 2:526-541, 2001.
13. Drew McDermott. *Nonmonotonic Logic II: nonmonotonic Modal Theories.* Journal of the Association for Computing Machinery, Vol 29 No. 1 January 1982. 33-57.
14. Pierluigui Minari. *A note on Łukasiewicz's three-valued logic.* Essay from Pierluigi Minari, Dept. of Philosophy, University of Florence. .
15. Mauricio Osorio, Verónica Borja, José Arrazola: Closing the Gap between the Stable Semantics and Extensions of WFS. *Mexican International Conference on Artificial Intelligence (MICAI) 2004* 202-211.
16. Mauricio Osorio, Juan Antonio Navarro, José Arrazola. "Applications of Intuitionistic Logic in Answer Set Programming", accepted in *Journal of TPLP,* 2003.
17. Mauricio Osorio, Juan Antonio Navarro, José Arrazola. "A Logical Approach to A-prolog.", 9th. Workshop on Logic Language and Information. Brazil, 2002.
18. David Pearce. Stable inference as intuitionistic validity. *The Journal of Logic Programming 38* 1999. 79-91.
19. John S. Schlipf. Formalizing a Logic for Logic Programming. *Annals of Mathematics and Artificial Intelligence 5.* (1992) 279-302.
20. Allen van Gelder, Kenneth A. Ross, and John S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM 38.*(1991) 620-650.

Answer Set Programming and S4

Mauricio Osorio and Juan Antonio Navarro

Universidad de las Américas, CENTIA,
Sta. Catarina Mártir, Cholula, Puebla, 72820 México
josorio@mail.udlap.mx

Abstract. We develop some ideas in order to obtain a nonmonotonic reasoning system based on the modal logic S4. As a consequence we show how to express the well known answer set semantics using a restricted fragment of modal formulas. Moreover, by considering the full set of modal formulas, we obtain an interesting generalization of answer sets for logic programs with modal connectives. We also depict, by the use of examples, possible applications of this inference system.

It is also possible to replace the modal logic S4 with any other modal logic to obtain similar nonmonotonic systems. We even consider the use of multimodal logics in order to model the knowledge and beliefs of agents in a scenario where their ability to reason about each other's knowledge is relevant. Our results clearly state interesting links between answer sets, modal logics and multi-agent systems.

1 Introduction

The stable model semantics, since its introduction in 1988 by Gelfond and Lifschitz [1], has been recognized as a novel contribution to the communities of nonmonotonic reasoning and logic programming. Research groups in several institutions are developing theory and applications related to this semantics.

This logic programming paradigm evolved into what is known today as Answer Set Programming (ASP). The basic idea of ASP is to provide a formal system that assigns to each *logic program,* the description of a problem, some set of *desirable models,* hopefully the problem solutions, called *answer sets.* The development efficient implementations of answer sets finders also allowed the creation of several applications that range from planning, solving combinatorial problems, verification, logical agent systems and product configuration.

Modal logic originated, on the other hand, as a consequence of the study of notions such as "necessary" and "possible". Extending the syntax of logic formulas with new unary connectives $\Box$ and $\Diamond$, and giving an adequate semantical meaning to them, it is possible to define formal systems to model knowledge, tense and obligation. Modal formulas usually have a very natural reading close to their intended intuitive meaning. This is one of the reasons why they had been quite useful to provide foundations for several applications in knowledge representation, multi-agent systems, etc.

In this paper we use the modal logic S4 in a general framework to model nonmonotonic reasoning. This is possible due to a characterization of answer sets in terms of intuitionistic logic that we have recently provided. This result, as well as several properties and consequences, are presented in [7–9]. Some other ideas relating modal logics and logic programs are also discussed in [6].

Moreover, we prove that the ASP approach is embedded in our proposed S4 nonmonotonic semantics. This is not really a surprise since, by virtue of the characterization of answer sets, intuitionistic logic can be embedded into modal logic S4 (thanks to a well known Gödel's translation).

We propose the following interpretation for our system: Consider a logic agent with some modal theory as its base knowledge. The agent could use the logic S4, a logic of knowledge, in order to do inference and produce new knowledge. However, we would also like our agent to be able to do nonmonotonic reasoning.

Informally speaking we will allow our agent to suppose some *simple acceptable knowledge* in order to make more inference. This simple acceptable knowledge consists of formulas of the form $\Box\Diamond F$, where F is a formula containing only unary connectives. Such formula could be read as: "the agent knows that the fact F is possible". If the agent can to justify all his assumptions (i.e. to prove all these formulas F) and obtain some sort of *complete* explanation for his base knowledge then we say it is safe for him to *believe* this new information.

The fact that nonmonotonic reasoning can be done via S4 has been already shown in [13]. Our approach is different and follows a line of research, originally proposed by Pearce [10], that tries to find relations among intuitionistic, modal logics and answer sets. Most of the results by Pearce were developed for disjunctive logic programs and, as one of the contribution of this paper, we present now some generalizations for propositional theories.

Our paper is structured as follows: In Section 2 we briefly introduce the syntax of propositional modal logic and the basics of S4. In Section 3 we present our framework for doing nonmonotonic reasoning using the logic S4. In Section 4 we introduce a previous result that relates answer sets with intuitionistic logic and, in Section 5, we establish the relations found with respect to the logic S4. Finally we present in Section 6 some ideas on how to develop an ASP approach for multi-agent systems formulating an example with two different agents. We finish with some conclusions in Section 7.

2 Background

In this section we briefly introduce some basic concepts and definitions that will be used along this paper. We introduce the language of propositional modal logic and the proof theory of the modal logic S4.

2.1 Propositional Modal Logic

We use the set of propositional modal formulas in order to describe rules and information within logic programs. Formally we consider a language built from an alphabet containing: a denumerable set $\mathcal{L}$ of elements called *atoms* or *atomic formulas;* the binary connectives $\wedge$, $\vee$ and $\rightarrow$ to denote conjunction, disjunction and implication respectively; the unary connective $\Box$ as a *knowledge operator;* the 0-ary connective $\bot$ to denote falsity; and auxiliary symbols (,).

Formulas can be constructed as usual in logic. The negation F can be introduced as an abbreviation of the formula $F \rightarrow \bot$, the *belief operator* $\Diamond F$ to abbreviate $\neg\Box\neg F$ and, similarly, the truth symbol $\top$ that stands for $\neg\bot$. We also can write, as usual, $F \leftrightarrow G$ to denote the formula $(F \rightarrow G) \wedge (G \rightarrow F)$. Finally, the formula $G \leftarrow F$ is just another way of writing $F \rightarrow G$.

A *modal theory,* or *modal program,* is a set of modal formulas, we restrict our attention however to finite theories. For a given theory T its *signature,* denoted $\mathcal{L}_T$, is the set of atoms that occur in the theory T. Observe that, since we consider finite theories, their signatures are also finite. Given a theory T we also define the negated set $\neg T = \{\neg F \mid F \in T\}$ and the knowledge set $\Box T = \{\Box F \mid F \in T\}$.

A *literal* is either a formula of the form a (positive literal) or $\neg a$ (negative literal) where a is an atom. Given a theory T we use $Lit_T = \mathcal{L}_T \cup \neg\mathcal{L}_T$ to denote the set of all literals that are relevant to T. If, for instance, we have the theory $T = \{\neg a \rightarrow b\}$ then $Lit_T = \{a, \neg a, b, \neg b\}$.

2.2 Modal Logic S4

Modal logic was originally conceived as the logic of necessary and possible. The logic S4 can be defined as the Hilbert type proof system that contains the following axiom schemes:

1. $(F \rightarrow (G \rightarrow H)) \rightarrow ((F \rightarrow G) \rightarrow (F \rightarrow H))$
2. $(F \rightarrow (G \rightarrow F)$
3. $\neg\neg F \rightarrow F$
4. $\Box(F \rightarrow G) \rightarrow (\Box F \rightarrow \Box G)$
5. $\Box F \rightarrow \Box\Box F$
6. $\Box F \rightarrow F$

and is closed under the rule of Necessitation (from F we can derive $\Box F$) as well as Modus Ponens (from F and $F \rightarrow G$ we can derive G). The behavior of other connectives follows from their usual definition in classical logic.

We use the standard notation $\vdash F$ to denote that F is a provable formula in the logic S4. If T is a theory we understand the symbol $T \vdash F$ to mean that $\vdash F_1 \wedge \cdots \wedge F_n \rightarrow F$ for some F_i contained in T. Similarly, given a theory U, we use the symbol $T \vdash U$ to denote $T \vdash F$ for every $F \in U$. A theory T is said to be consistent, with respect to the logic S4, if it is not the case that $T \vdash \bot$. We use the notation $T \Vdash U$ to stand for the phrase: T is consistent and $T \vdash U$.

3 Non-monotonic Reasoning in S4

In this section, we introduce our proposed nonmonotonic inference system using S4, in particular the notion of a *weakly complete and consistent extension* for a given theory. The idea of our approach is as follows: *"If we cannot derive a formula F by a standard inference in S4 from a theory T, we could try to derive the formula F by a 'suitable' extension of the theory T."*

The first basic requirement of this extended theory is to be consistent. Second, the extra formulas that we include should be a sort of 'weak' assumptions. Any formula of the form $\Box\Diamond F$ (where F is any formula containing only 1-place connectives) is considered a 'weak' assumption. Such formula just says that it is known that is possible something. As noted before, we borrow such formula if it helps us to obtain a consistent explanation of the world.

Definition 1. *Let P be any modal theory and let* $M \subseteq Lit_P$*. The* modal closure *of M is defined as* $\overline{M} = \neg(Lit_P \setminus M) \cup \Box M$*. Then* $\Box M$ *is an* S4-answer set *of the theory P if* $P \cup \Box\Diamond\overline{M} \Vdash_{S4} \Box M$*. Moreover* $P \cup \Box\Diamond\overline{M}$ *is called a* weakly complete and consistent extension *of P.*

Consider the program $P = \Box(\neg\Box a \rightarrow b)$ with $Lit_P = \{a, \neg a, b, \neg b\}$. Observe that $\{\Box b\}$ is an S4-answer since $P \cup \Box\Diamond\{\neg\neg a, \neg a, \neg\neg b, \Box b\}$ is consistent and proves, under S4, the formula $\Box b$. This program has no more answer sets. We could also consider the following more interesting example:

1. Juan is mexican.
2. Mary is american.
3. Pablo is mexican and not catholic.
4. It is known that normally mexicans are catholic.

We can encode this problem using the following program:

$\Box(mexican(juan))$.
$\Box(american(mary))$.
$\Box(mexican(pablo) \wedge \neg catholic(pablo))$.
$\Box((mexican(X) \wedge \neg\Box\neg catholic(X)) \rightarrow catholic(X))$.

Note that the sentence "It is known that normally mexicans are catholic" is encoded by the last rule, that says: It is known that 'if it is not known that *X* is not catholic' and '*X* is mexican' then '*X* is catholic'. Of course, we also need (not just S4) our extended nonmonotonic S4 to make this example work as we immediately explain. The unique s4-answer set of this program is:

$$\Box\,\{\neg catholic(pablo), mexican(pablo), mexican(juan), \\ catholic(juan), american(mary)\}$$

So, we know that Pablo is not catholic. We also know that Juan is catholic. However, this knowledge comes from nonmonotonic reasoning. This kind of knowledge, not derived by the regular inference procedure of S4, can be considered as a "weak knowledge" or a "strong belief" so to speak. The S4-answer set does not decide whether Mary is catholic or not, since the program does not provide any clue about this fact.

Why modal logic S4? The S4 and S5 systems are perhaps the two most well known systems to represent the notions of knowledge and possibility. The theorem that we present in the next section however fails if we use S5 instead of S4. The problem is that S5 has no "irreducible iterated modalities". In S5, $\Box F$ is equivalent to $\Diamond\Box F$. We need, however, to distinguish these two formulas in order to gain expressibility. Many other logics, between S4 and S5, behave nicely with respect to our approach.

4 Expressing Answer Sets

The answer set semantics is a popular semantic operator for logic programs. One of the main features of answer sets is the introduction of negation as failure which is extremely useful to model notions such as nonmonotonic reasoning, default knowledge and inertial rules. The definition of answer sets is not required for the purposes of this paper, the reader is referred to [8] for more details. It is just important to mention that answer sets are defined for augmented logic programs, a class of propositional logic programs (without modal connectives), and incorporates an additional *classical negation* connective, denoted ~, that can only be used preceding atomic occurrences.

4.1 Logical Foundations of A-Prolog

The characterization of answer sets in terms of intermediate logics is an important result that provides solid logical foundations to this paradigm. Pearce initiated this line of research using intuitionistic extensions, obtained by adding negated atoms, to characterize answer sets for disjunctive logic programs [12]. This procedure, however, is not able to obtain the answer sets of logic programs containing negation in the head [8].

Alternatively Pearce developed another approach using extensions of theories based on the logic HT, and showed that they are equivalent to the *equilibrium logic* also formulated by himself [11, 12]. In a more recent paper [5], together with Lifschitz and Valverde, Pearce was able to show that equilibrium models can be used to obtain the answer sets of augmented logic programs.

Following the original idea from Pearce we were able to show that a characterization of answer sets for augmented programs is also possible in terms of intuitionistic logic. We do consider extensions with negated atoms, as Pearce did, but also allow double negated atoms ($\neg\neg a$) in our intuitionistic extensions. Pearce itself suggested how modal logics could be used to model answer sets for augmented programs as a consequence of his results on Nelson's logic. Our contribution in this paper is to fill in the details and explore possible advantages of this approach. The following theorem was stated and proved in [8].

Theorem 1. *Let P be an augmented program and let $M \subseteq \mathcal{L}_P$. M is an answer set of P if and only if $P \cup \neg(\mathcal{L}_P \setminus M) \cup \neg\neg M \Vdash_{\mathrm{I}} M$.*

Previous theorem assumes that augmented programs do not contain classical negation, recall that the role of classical negation is not defined in the context of intuitionistic logic. This assumption, however, does not impose any important restriction since classical negation can be easily simulated as will be explained in Section 4.2. On the other hand, this result suggests a natural way to extend the notion of answer sets for any propositional theory T. The idea was also proposed in [8] and used later to develop the *safe belief semantics* in [9].

Definition 2. *Let P be any propositional theory and let $M \subseteq \mathcal{L}_P$. We define M to be an answer set of P if and only if $P \cup \neg(\mathcal{L}_P \setminus M) \cup \neg\neg M \Vdash_{\mathrm{I}} M$.*

We have also shown that the answer set semantics is invariant under any intermediate logic and generalized this approach to include extensions of the form $\neg\neg F$, with F any formula, providing a more general framework to study and define semantics, see [9]. These results are (in our point of view) good evidence of the well-behavior of the answer set semantics in logical terms.

4.2 Restoring Classical Negation

We will now show how to restore the use of classical negation in our logic programs. Recall that ~ is only allowed preceding atomic formulas so that, intuitively, we may think of the formula $\sim a$ just as an atom with a convenient name so that an answer set finder can discard models where both a and $\sim a$ appear.

Formulas of the form a and $\sim a$ are referred as *~literals,* and we say that a set of ~literals M is consistent if it is not the case that both a and $\sim a$ are contained in M for some atom a. We will also use the terms *asp-formulas, asp-theories* and *asp-programs* to denote entities that allow the use of classical negation. We also define, if M is a set of atoms, $\sim M = \{\sim a \mid a \in M\}$. Moreover, for an asp-program P its *extended signature* will be $\mathcal{L}_P^{\sim} = \mathcal{L}_P \cup \sim\mathcal{L}_P$.

Definition 3. *Given a signature $\mathcal{L}$, let $\mathcal{L}'$ be another signature with the same cardinality as $\mathcal{L}$ and with $\mathcal{L} \cap \mathcal{L}' = \emptyset$. Also let $f\colon \mathcal{L} \to \mathcal{L}'$ be a biyective function between the two signatures. We define the mapping + from asp-formulas over the signature $\mathcal{L}$ to formulas over $\mathcal{L} \cup \mathcal{L}'$ recursively as follows:*

1. *$(\bot)^+ = \bot$.*
2. *for any atom a let $(a)^+ = a$ and $(\sim a)^+ = f(a)$.*
3. *for any pair of formulas F, G let $(F \odot G)^+ = F^+ \odot G^+$ where $\odot \in \{\wedge, \vee, \to\}$.*

The definition of + is also extended to theories as usual, $T^+ = \{F^+ \mid F \in T\}$.

Lemma 1. *Let P be an augmented asp-program and let $M \subseteq \mathcal{L}_P^{\sim}$ be a consistent set of ~literals. Let + be a mapping defined with the corresponding set $\mathcal{L}'_P$ and a function $f\colon \mathcal{L}_P \to \mathcal{L}'_P$. M is an answer set of P if and only if M^+ is an answer set of P^+.*

Proof. Follows as a direct generalization of Proposition 2 in [2].

Theorem 2. *Let P be an augmented asp-program, and let M be a consistent set of ~literals. M is an answer set of P iff $P^+ \cup \neg(\mathcal{L}_{P^+} \setminus M^+) \cup \neg\neg M^+ \Vdash_{\mathrm{I}} M^+$.*

Proof. Follows immediately by Theorem 1 and previous lemma.

5 Characterization of Answer Sets Using S4

The purpose of the following translation, given in [3], is to provide a meaning to any propositional theory including classical negation with a broader role, and not only for the class of augmented programs. We can consider, for instance, the use of the classical negation connective for any arbitrary formula and not only for single atoms. We will see that propositional modal formulas are expressive enough to model the two kinds of negations in asp-programs.

Definition 4. *The translation* $^\circ$ *of asp-formulas to modal formulas is defined recursively as follows:*

$$\begin{array}{ll}
(a)^\circ = \Box a, \quad \textit{for atomic } a & (a)^* = \Box\neg a, \quad \textit{for atomic } a \\
(F \vee G)^\circ = F^\circ \vee G^\circ & (F \vee G)^* = F^* \wedge G^* \\
(F \wedge G)^\circ = F^\circ \wedge G^\circ & (F \wedge G)^* = F^* \vee G^* \\
(F \to G)^\circ = \Box(F^\circ \to G^\circ) & (F \to G)^* = F^\circ \wedge G^* \\
(\neg F)^\circ = \Box\neg F^\circ & (\neg F)^* = F^\circ \\
(\sim F)^\circ = F^* & (\sim F)^* = F^\circ
\end{array}$$

The definition is also extended to sets of asp-formulas, $T^\circ = \{F^\circ \mid F \in T\}$.

Observe that our translation behaves just like the Gödel's translation of intuitionistic logic into S4 for programs without classical negation. Due to the well known Gödel embedding of intuitionistic logic in S4 and Theorem 2, we can express answer sets for augmented programs, where the role of classical negation is "passive" (because it is only applied to atoms).

Theorem 3. *Let P be an augmented asp-program, and let M be a consistent set of ~literals. The set M is an answer set of the logic program P if and only if* $(P^+ \cup \neg(\mathcal{L}_{P^+} \setminus M^+) \cup \neg\neg M^+)^\circ \Vdash_{S4} M^{+^\circ}$.

In the rest of this section we discuss an alternative representation of answer sets in order to model an active role of classical negation.

Definitions 5. *Let* $\mathcal{L}$ *and* $\mathcal{L}_1$ *two disjoint signatures and let* f *a bijective function from* $\mathcal{L}$ *to* $\mathcal{L}_1$. *We define a mapping —from modal formulas with signature* $\mathcal{L} \cup \mathcal{L}_1$ *to formulas in* $\mathcal{L}$ *recursively as follows:*

1. $(\bot)^- = \bot$.
2. *for any atom a let* $(a)^- = a$ *if* $a \in \mathcal{L}$ *and* $(a)^- = \neg f^{-1}(a)$ *otherwise.*
3. *for any pair of formulas F, G let* $(F \odot G)^- = F^- \odot G^-$ *where* $\odot \in \{\wedge, \vee, \to\}$.
4. *for any formula F let* $(\Box F)^- = \Box F^-$.

We also extend our translation over theories: $T^- = \{F^- \mid F \in T\}$.

Lemma 2. *If P is an augmented asp-program, defined over* $\mathcal{L}$, *then* $P^{+^{\circ^-}} = P^\circ$.

Proof. Without lost of generality it suffices to consider a singleton program. Then, it suffices to apply a direct induction on the size of the formula.

Theorem 4. *Let P be an augmented asp-program, and let M be a consistent set of ~literals. Then M is an answer set of P iff* M° *is a* S4*-answer set of* P°.

Proof. Let $\mathcal{L}'_P$ be a new signature with the same cardinality as $\mathcal{L}_P$ and such that $\mathcal{L}'_P \cap \mathcal{L}_P = \emptyset$. Let $f\colon \mathcal{L}_P \to \mathcal{L}'_P$ be a bijective function. We have then that a consistent set of ~literals M is an answer set of P:

iff $P^+ \cup \neg(\mathcal{L}_{P^+} \setminus M^+) \cup \neg\neg M^+ \Vdash_{\mathrm{I}} M^+$, by Theorem 2.
iff $(P \cup \neg(\mathcal{L}_P^{\sim} \setminus M) \cup \neg\neg M)^+ \Vdash_{\mathrm{I}} M^+$, by the definition of the mapping +.
iff $(P \cup \neg(\mathcal{L}_P^{\sim} \setminus M) \cup \neg\neg M)^{+^\circ} \Vdash_{\mathrm{S4}} M^{+^\circ}$, by Gödel's embedding of I into S4.
iff $(P \cup \neg(\mathcal{L}_P^{\sim} \setminus M) \cup \neg\neg M)^{+^{\circ^-}} \Vdash_{\mathrm{S4}} M^{+^{\circ^-}}$, by a proof by induction on the length of the S4 proof and since M is a consistent set of ~literals (the very restricted class of formulas added as an extension is also required).
iff $(P \cup \neg(\mathcal{L}_P^{\sim} \setminus M) \cup \neg\neg M)^\circ \Vdash_{\mathrm{S4}} M^\circ$, by Lemma 2.
iff $P^\circ \cup \Box\Diamond(\neg(Lit_{P^\circ} \setminus M^*) \cup M^\circ) \Vdash_{\mathrm{S4}} M^\circ$, by set theory and definition of $\circ$, where M^* is obtained from M replacing ~ with $\neg$. Note that $M^\circ = \Box M^*$, also that $(\neg S)^\circ = \Box\Diamond\neg S^*$ and $(\neg\neg S)^\circ = \Box\Diamond\Box S^*$ for any set of ~literals S.
iff M° is an s4-answer set of P°, by the definition of S4-answer sets.

The following theorem is one of the main contributions of this paper. It shows, thanks to the invariance of the answer set semantics with respect to intermediate logics proved in [9], that also a wide family of modal logics can be used to characterize answer sets using the current approach.

Theorem 5. *Let P be an augmented asp-program, and let M be a consistent set of ~literals. Then M is an answer set of P iff* M° *is a* X-answer *set of* P°. *Where X is any logic between S4 and S4.3 inclusive.*

Proof. Follows from Theorem 4 and results in [9].

The transformations proposed and studied in [3] exhibit, in particular, an embedding of the logic of Nelson N into S4. Recall that the logic of Nelson already considers two kind of negations which correspond to the - and ~ we introduced here. We believe that, based on the results presented on [3], the proofs of theorems presented in this section could be simplified. It would also make the introduction of ~ less artificial.

6 ASP for Multimodal Logic

ASP for multimodal logic would require to generalize Definition 1. Instead we just introduce here an example and leave a formal presentation for a future paper. We will consider *"The Wise Man Puzzle"* to show how nonmonotonic reasoning can be used to model the knowledge and beliefs of agents where interaction between them has an important role[1]. This puzzle is typically stated as follows [4]:

[1] We would like to emphasise that we are not trying to model the behavior or capabilities of *agents,* instead we just try to show how our nonmonotonic system could provide a convenient way to represent the *knowledge* of these agents and the way that their knowledge interacts.

A king wishes to determine which of his three wise men is the wisest. He arranges them in a circle so that they can see and hear each other and tells them that he will put a white or a black spot on each of their foreheads, and that at least one spot will be white. In fact all three are white. He offers his favor to the one who can tell him the color of his spot. After a while, the wisest announces that his spot is white. How does he Know?

The solution is based, of course, on the ability of the agents involved to reason about knowledge and beliefs of other agents, information that can be observed and the rules stated by the king at the beginning of the game. We will model a shorter and simpler version in which only two wise man participate. We write w_1 (w_2) to denote that the first (second) wise man has a white spot on his forehead. We can state a set of assumptions P as follows:

$$\begin{array}{ll}
\Box_1(w_1 \vee w_2), \Box_2(w_1 \vee w_2), & (1)\\
\Box_1\Box_2(w_1 \vee w_2), \Box_2\Box_1(w_1 \vee w_2), & \\
\Box_1(w_1 \rightarrow \Box_2 w_1), \Box_2(w_2 \rightarrow \Box_1 w_2), & (2)\\
\Box_1(\neg w_1 \rightarrow \Box_2\neg w_1), \Box_2(\neg w_2 \rightarrow \Box_1\neg w_2), & \\
\Box_2\Diamond_2\Diamond_1 w_1 \rightarrow \Box_2\Diamond_1 w_1, & (3)\\
\Box_2\Diamond_2\Diamond_1\neg w_1 \rightarrow \Box_2\Diamond_1\neg w_1, &
\end{array}$$

The group of rules under block (1) state the fact that each wise man knows the king's announcement: *"at least one spot will be white"*, and they also know each other knows this information. The rules in block (2) state that they know that they can see each other. Finally the pair of rules under (3) are the nonmonotonic part of our program. Those rules will make the second wise man assume, if it makes sense, that the first wise man believes w_1 (or $\neg w_1$).

Note that $P \nvdash_{S4} \Box_2 w_2$ since, in principle, the second wise man can not know *for sure,* using only the information in P, the fact that he has a white spot. He cloud try, however, to do nonmonotonic inference extending his theory with simple acceptable knowledge. Adding $\Box_2\Diamond_2\Diamond_1\neg w_1$ to P allows him, in fact, to prove that $\Box_2 w_2$. If we take the set $M = \Box_2\Diamond_2\{\Box_2 w_1, \Box_2 w_2, \Diamond_1 w_1, \Diamond_1\neg w_1, \Box_1 w_2\}$ as a set of simple acceptable knowledge, it turns out that $P \cup M$ is consistent and proves the facts $\{\Box_2 w_1, \Box_2 w_2, \Box_1 w_2\}$. A reasonable definition of ASP for multimodal logic should recognise this set as a valid answer set.

As an important observation notice that we did not have to explicitly include $w_1 \wedge w_2$ as a fact in the program P. This is interesting since agents are reasoning about the world without depending on the actual situation going on, and thus leaving a more general program. Suppose for instance that only the first wise man has a white spot (i.e. $w_1 \wedge \neg w_2$) then he could deduce, just after seeing his partner, $\Box_1 w_1$. The model M discussed above is no longer consistent with $P \cup \{\Box_1 w_1\}$ and, therefore, it should not be answer set.

This is exactly the notion of nonmonotonic reasoning we are trying to model. It is, in some sense, safe for the second wise man to assume he has a white spot. Until he has evidence to believe the opposite, that is when the first wise man makes explicit his knowledge about the situation of the world, then he can provide an answer for sure. What we gain is the possibility to start making inference about the knowledge or beliefs of other agents without needing them to explicitly state such information.

Theorem proving tools of the Logics Workbench LWB[2], developed at the University of Bern in Switzerland, were used to check the proofs in this section.

7 Conclusions

We propose how to do nonmonotonic reasoning using the modal logic S4. We also showed, generalizing a previous result in intuitionistic logic, how we can express the well known answer sets semantics using our approach. Observe that, in principle, it is possible to replace S4 with other stronger logics, up to S4.3, to get similar nonmonotonic systems. Interesting applications can also emerge if we allow the use of multimodal logic to model several interacting agents aimed with nonmonotonicity. Our results clearly state interesting links between ASP, modal and multi-modal systems, which might bring research of these areas together.

References

1. Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In R. Kowalski and K. Bowen, editors, *5th Conference on Logic Programming,* pages 1070–1080. MIT Press, 1988.
2. Michael Gelfond and Vladimir Lifschitz. Logic programs with classical negation. In David H. D. Warren and Péter Szeredi, editors, *Logic Programming, Proceedings of the Seventh International Conference,* pages 579–597, Jerusalem, Israel, June 1990. MIT Press.
3. Jan Jaspars. *Calculi for Constructive Communication, A study of the dynamics of partial states.* ITK and ILLC dissertation series, 1994.
4. Kurt Konolige. *A Deduction Model of Belief.* Morgan Kaufman Publishers, Inc., 1986.
5. Vladimir Lifschitz, David Pearce, and Agustin Valverde. Strongly equivalent logic programs. *ACM Transactions on Computational Logic,* 2:526–541, 2001.
6. Mauricio Osorio and Juan Antonio Navarro. Modal logic $S5_2$ and FOUR (abstract). In *2003 Annual Meeting of the Association for Symbolic Logic,* Chicago, June 2003.
7. Mauricio Osorio, Juan Antonio Navarro, and José Arrazola. A logical approach for A-Prolog. In Ruy de Queiroz, Luiz Carlos Pereira, and Edward Hermann Haeusler, editors, *9th Workshop on Logic, Language, Information and Computation (WoLLIC),* volume 67 of *Electronic Notes in Theoretical Computer Science,* pages 265–275, Rio de Janeiro, Brazil, 2002. Elsevier Science Publishers.
8. Mauricio Osorio, Juan Antonio Navarro, and José Arrazola. Applications of intuitionistic logic in answer set programming. *Theory and Practice of Logic Programming,* 4(3):325–354, 2004.
9. Mauricio Osorio, Juan Antonio Navarro, and José Arrazola. Safe beliefs for propositional theories. Accepted to appear at Annals of Pure and Applied Logic, 2004.

[2] http://www.lwb.unibe.ch/

10. David Pearce. Answer set inference and nonmonotonic S4. In *Extensions of Logic Programming,* number 798 in Lecture Notes in Artificial Intelligence. Springer, Berlin, 1994.
11. David Pearce. From here to there: Stable negation in logic programming. In D. M. Gabbay and H. Wansing, editors, *What Is Negation?,* pages 161–181. Kluwer Academic Publishers, Netherlands, 1999.
12. David Pearce. Stable inference as intuitionistic validity. *Logic Programming,* 38:79–91, 1999.
13. Grigori Schwarz and Miroslaw Truszczynski. Nonmonotonic reasoning is sometimes simpler! *Journal of Logic and Computation,* pages 295–308, 1996.

A Rippling-Based Difference Reduction Technique to Automatically Prove Security Protocol Goals*

Juan Carlos López and Raúl Monroy

Computer Science Department, ITESM—Estado de México,
Carretera al lago de Guadalupe, Km. 3.5, Estado de México, 52926, Mexico
{juan.pimentel, raulm}@itesm.mx

Abstract. The inductive approach [1] has been successfully used for verifying a number of security protocols, uncovering hidden assumptions. Yet it requires a high level of skill to use: a user must guide the proof process, selecting the tactic to be applied, inventing a key lemma, etc. This paper suggests that a proof planning approach [2] can provide automation in the verification of security protocols using the inductive approach. Proof planning uses AI techniques to guide theorem provers. It has been successfully applied in formal methods to software development. Using inductive proof planning [3], we have written a method which takes advantage of the differences in term structure introduced by rule induction, a chief inference rule in the inductive approach. Using difference matching [4], our method first identifies the differences between a goal and the associated hypotheses. Then, using rippling [5], the method attempts to remove such differences. We have successfully conducted a number of experiments using HOL-Clam [6], a socket-based link that combines the HOL theorem prover [7] and the Clam proof planner [8]. While this paper key's contribution centres around a new insight to structuring the proof of some security theorems, it also reports on the development of the inductive approach within the HOL system.

1 Introduction

The inductive approach [1] articulates the verification of security protocols in a formal setting. The inductive approach has been widely used for verifying protocols, but it requires a high level of skill to use. A user must guide the proof process: selecting the subgoaling strategies (called *tactics*) to be applied, inventing key lemmas, etc. Not only are the proofs deep but also they are onerous and cumbersome.

Proof planning is a meta-level reasoning technique, developed especially to automate theorem proving [2]. A proof plan expresses the patterns of reasoning shared by members of the same family of proofs, and is used to drive the search

* This research was partially supported by two grants: CONACYT 33337-A and ITESM CCEM-0302-05.

for new proofs in that family. Proof planning works by using formalised pre- and post- conditions of tactics as the basis of plan search. These high-level specifications of tactics are called *methods.* Proof planning has been successfully tested in a number of domains, including software verification [9].

This paper suggests that a proof planning approach can provide automation in the verification of authentication protocols using Paulson's approach. It reports on a few steps taken towards this aim. Using a proof plan for induction [3], we have written a method which takes advantage of the differences in term structure that are introduced by an application of rule induction. Being the reason for the name, rule induction is a chief inference rule in the inductive approach for the verification of security protocols. Our method first identifies the differences between a goal and its associated hypotheses, applying difference matching, and then removes them, applying rippling [5], core of the inductive proof plan.

We have successfully tested our method on a number of security goals. We have conducted our experiments on HOL-Clam [6], a socket-based link that combines the HOL theorem prover [7] and the Clam proof planner [8]. While this paper key's contribution centres around a new insight to structuring the proof of some security theorems, it also reports on the development of the inductive approach within the HOL system.

The rest of this paper is organised as follows: Section 2 outlines the inductive approach to verification, emphasising key proof steps and major difficulties in proof discovery. Section 3 describes proof planning, the approach we have adopted to automate theorem proving. In particular, Section 3 describes rippling and difference matching, two techniques central to our method, which is described in Section 4. Section 5 indicates directions for further work and details drawn conclusions.

2 The Inductive Approach: An Overview

In the inductive approach, a *protocol* is circumscribed as the set of all possible traces that it can take. A *protocol trace* is a list of communicating events. There are four types of *events:*

Says $A\ B\ M$, which means that agent A has sent message M to agent B;
Gets $A\ M$, which means that agent A has received message M;[1]
Notes $A\ M$, which models that agent A has performed a computation over the content of message M and then stored the result; and
Oops, which models that an agent has accidentally lost a piece of critical information, such as an session key.

An attack is modelled directly by modifying the protocol specification, imposing unfaithful participations of one or more agents.

The model involves three sorts of agents: the server, S, the spy, Spy, and the friendly agents, $A, B, \ldots.$ The server is absolutely trusted. However, friendly

[1] Notice that the originator of the message is not taken for granted.

agents may get compromised; in symbols, $A \in \mathsf{bad}$. Only the spy may read messages not intended to himself. He holds the long-term key of all compromised agents, hence possibly posing as them, but may also send genuine messages using his own long-term key.

2.1 Messages and Message Analysis

Messages comprise agent names, S,Spy, *A*, *B*,..., fresh labels (called *nonces*), $N_a, N_b, \ldots$, shared keys, $K_{as}, K_{bs}, \ldots$, public keys, $K_a, K_b, \ldots$, private keys, $K_a^{-1}, K_b^{-1}, \ldots$, session keys, $K_{ab}, K_{ac}, \ldots$, compound messages, $\{|X, Y|\}$, and messages encrypted under key *K*, $\{|X|\}_K$. By convention, an encrypted message can neither be read nor altered without the corresponding encryption key.

The theory of messages involves three main operators. Each operator is defined on possibly infinite sets of messages and is used to model properties of a protocol and reason about its runs. Let *H* be a set of messages, then the operators are given as follows:

parts *H* returns all the message components that can be recursively extracted from messages in *H* by projection and decryption;
analz *H* is as parts *H*, except that decryption of traffic is performed only using available keys; and
synth *H* models the messages that the spy can forge using only *H* and available keys.

2.2 Events

The theory of messages is at the bottom of a 3-layer, hierarchical model. Building on the theory of messages, the theory of events is at the next upper layer. The event theory aims at characterising the knowledge an agent may reach upon a run of a protocol. Additionally, it formalises a notion of message freshness, which is useful to separate past and present. There are three key relation symbols defined in events:

1. initState *A*, which models the initial knowledge of agent *A*;
2. knows *A evs*, which models what agent *A* can learn from a protocol trace, *evs*, considering *A*'s initial knowledge; and
3. used *evs*, which allows one to determine whether or not an element is fresh with respect to protocol trace *evs*.

2.3 Shared-Key and Public-Key Cryptography

At the top of the hierarchy, depending on the kind of protocol under analysis, there is either of two theories: shared, for protocols involving shared-key cryptography, and public, for protocols involving public-key and possibly shared-key cryptography.

Similar to Schneider's [10], Paulson's approach considers security properties from a high-level view. Thus, the identity of the spy is known and may be used to express properties. For example, let *evs* denote a protocol run, then $X \in$ analz (knows Spy *evs*) holds only if X is in the traffic, or if it is part of a compound message or if it is in the body of a message encrypted under a key known to the spy. As another example property, consider $X \in$ parts (knows Spy *evs*), which is used to specify that X is part of the traffic and so it has been issued, possibly inside a larger, encrypted message.

The analysis of a protocol proceeds by rule induction. It involves the use of over 400 properties, either (in)equalities or implications. These properties refer only to the inductive approach. Subsidiary results, imported from ancestor theories, for example, lists, sets, arithmetic, and so on, are also required.

A protocol is described as a collection of inference rules. Each inference rule has zero or more hypotheses (enclosed by square brackets, with semicolons separating them) and one conclusion, joined by $\vdash$. The inference rules state the various forms in which a protocol trace can be possibly extended with new events. Fig. 2.3 deploys a partial description of the Otway-Rees protocol. There, # and set respectively stand for function list constructor and the function that takes a list and then converts it into a set.

Rule Name	Definition
Nil	$[\,] \vdash [\,] \in \mathsf{otway}$
Fake	$[\mathsf{ef} \in \mathsf{otway}; X \in \mathsf{synth}\ (\mathsf{analz}\ (\mathsf{knows\ Spy\ ef}))]$ $\vdash \mathsf{Says\ Spy}\ B\ X \# \mathsf{ef} \in \mathsf{otway}$
Reception	$[\mathsf{er} \in \mathsf{otway}; \mathsf{Says}\ A\ B\ X \in \mathsf{set\ er}]$ $\vdash \mathsf{Gets}\ B\ X \# \mathsf{er} \in \mathsf{otway}$
OR1	$[\mathsf{e1} \in \mathsf{otway}; \mathsf{Nonce}\ NA \notin \mathsf{used\ e1}]$ $\vdash \mathsf{Says}\ A\ B\ \{\lvert \mathsf{Nonce}\ NA, \mathsf{Agent}\ A, \mathsf{Agent}\ B,$ $\{\lvert \mathsf{Nonce}\ NA, \mathsf{Agent}\ A, \mathsf{Agent}\ B \rvert\}_{K_{AS}} \rvert\} \# \mathsf{e1} \in \mathsf{otway}$
OR2	$[\mathsf{e2} \in \mathsf{otway}; \mathsf{Nonce}\ NB \notin \mathsf{used\ e2};$ $\mathsf{Gets}\ B\ \{\lvert \mathsf{Nonce}\ NA, \mathsf{Agent}\ A, \mathsf{Agent}\ B, X \rvert\} \in \mathsf{set\ e2}]$ $\vdash \mathsf{Says}\ B\ \mathsf{S}\ \{\lvert \mathsf{Nonce}\ NA, \mathsf{Agent}\ A, \mathsf{Agent}\ B, X,$ $\{\lvert \mathsf{Nonce}\ NA, \mathsf{Nonce}\ NB, \mathsf{Agent}\ A, \mathsf{Agent}\ B \rvert\}_{K_{BS}} \rvert\} \ldots$
⋮	⋮
OR4 ...	
Oops	$[eo \in \mathsf{otway}; \mathsf{Says\ S}\ B \{\lvert \mathsf{Nonce}\ NA, X, \{\lvert \mathsf{Nonce}\ NB, \mathsf{Key}\ K \rvert\}_{K_{BS}} \rvert\} \in \mathsf{set}\ eo]$ $\vdash \mathsf{Notes\ Spy}\ \{\lvert \mathsf{Nonce}\ NA, \mathsf{Nonce}\ NB, KeyK \rvert\} \# eo \in \mathsf{otway}$

Fig. 1. (Partial) Inductive definition of the Otway-Rees protocol

2.4 Isabelle

The inductive approach has been implemented in Isabelle/HOL, the Higher-Order Logic instantiation of the Isabelle generic theorem prover [11]. Isabelle contains a large collection of assorted proof methods, including a simplifier and several classical reasoners. These methods are all powerful and they all accept a number of switches, with which one can extend their scope of application.

Isabella also contains a variety of proof support tools, with which one can easily define new symbols, set a suitable, succinct syntax, and so on.

Despite Isabelle's powerfulness, using Isabelle to verify a security protocol demands a high level of skill. The proofs are so deep, onerous and so cumbersome that applying a proof method without using any switch is most likely not to be enough. Selecting the right switch to a proof method requires mastership on using Isabelle. For example, for blast[2] to work smoothly, a developer is required to carefully select which properties are to be used as elimination, introduction or destruction rules. She is also required to know the content of the classical rule sets in order to properly use them whenever necessary. In a similar vein, a developer is required to carefully select which properties are to be used for simplification. So, she constantly runs the risk of non-termination or having to work harder to find more proofs. Thus the inductive approach to protocol verification poses lots of challenges to full automation. This paper suggests that a proof planning approach can improve the existing level of automation in this domain, or at least extend it so as to take user-level interaction to a more adequate, convenient level.

3 Proof Planning

Proof planning approaches automatic theorem proving in two stages. One for building a proper proof plan to a given goal, and another for executing the plan to obtain a proof of the goal.

3.1 Methods

Methods are the building-blocks of proof planning. A *method* is a high-level description of a tactic, containing an input sequent, preconditions, output sequents, and effects or postconditions. A method is said to be *applicable* if the current goal matches the method input sequent and the method preconditions hold. Preconditions specify properties of the input sequent, with which proof planning predicts if the tactic associated with the method is applicable without actually running it, and likewise for the postconditions. The result of a method application is a list of output sequents, possibly empty.

The proof planner develops the plan by selecting a method applicable to the current goal. If no method is applicable, it will terminate, reporting failure. Otherwise, the proof planner will give consideration to the subgoals returned by the first applicable method. This process is applied recursively to each subgoal till no more methods are applicable, or, as in the normal case of success, all the leaves of the proof plan tree return an empty list of output sequents.

Inductive proof planning is the application of proof planning to automating inductive theorem proving [3]. It involves the use of *rippling* [5], a heuristic that guides the search for a proof of inductive cases and supports the selection of appropriate induction schemata. Rippling guides the manipulation of the induction

[2] Blast is Isabelle's main workhorse and one of the classical reasoners.

conclusion to enable the use of a hypothesis, called *fertilization.* The key idea behind rippling lies in the observation that an initial induction conclusion is a copy of one of the hypotheses, except for extra terms. By marking such differences explicitly, rippling can attempt to place them at positions where they no longer preclude the conclusion and hypothesis from matching. Rippling applies a special kind of rewrite rules, called *wave-rules,* which manipulate the differences between two terms *(wave-fronts),* while keeping their common structure *(skeleton)* intact.

Annotated terms, called *wave-terms,* e.g. $\boxed{ev\#\underline{evs}}$, are composed of a wave-front, and one or more wave-holes. *Wave-fronts,* e.g. $\boxed{ev\#}$, are expressions that appear in the induction conclusion but not in the induction hypothesis. Inversely, *wave-holes,* e.g. $\underline{evs}$, are expressions that appear in wave-terms and also in the induction hypothesis.

3.2 Difference Unification

Wave-annotations are introduced by applying difference unification [4] to the hypotheses and the conclusion. *Difference unification* extends unification so that differential structures between the terms to be unified can also be hidden, while computing a substitution.

Ground difference unification is as difference unification except that it is restricted only to ground terms and is used to automate the dynamic generation of wave-rules. *Ground difference matching* is one-way ground difference unification and is used to distinguish term structural differences with wave-annotations, in order to serve rippling.

The inductive proof plan consists of 4 compound methods: i) `base_case`, ii) `normalize`, iii) `generalise` and iv) `ind_strat`. A *compound method* is a method that calls other, possibly compound, methods from its pre- or post-conditions.

4 Difference Reduction in Protocol Verification

We approach the problem of automatically verifying security goals by using two systems. One is *Clam* [8], a proof planner, and the other *HOL* [7], a theorem prover for Higher-Order logic. From a Clam's perspective, our method consists of an straightforward application of the inductive proof plan. It makes use of the four inductive proof methods and adds a new one, called `difference_reduction`. `difference_reduction` is as `step_case`, except that it first attempts to annotate the goal at hand and, if successful, removes them using the standard heuristic knowledge underlying `step_case`. `difference_reduction` annotates a sequent by ground difference matching the goal against the hypotheses. It takes advantage of the differences in term structure introduced by rule induction.

From a HOL's perspective, our method is just a powerful tactic, called `ONCE_CLAM_TAC`. The user actually deals with protocol verification through HOL, using the implementation of the inductive approach that we have already developed.[3] When facing a conjecture, the user simply calls `ONCE_CLAM_TAC` via HOL-Clam [6], a socket-based link that combines the HOL theorem prover and the Clam proof planner. In the normal case of success, `ONCE_CLAM_TAC` executes the tactic returned by Clam, which, furthermore, proves the goal or makes substantial progress towards proving it.

We illustrate our approach using a running example. Suppose that we want to prove that for Otway-Rees, see Fig. 2.3, an agent's long term key, $\mathsf{Key}\ (\mathsf{shrK}\ A)$, is on the network traffic if and only if that agent is compromised; in symbols:

$$evs \in \mathsf{otway} \longrightarrow (\forall A.\ \mathsf{Key}\ (\mathsf{shrK}\ A) \in \mathsf{parts}\ (\mathsf{knows}\ \mathsf{Spy}\ evs) = A \in \mathsf{bad}\ ) \qquad (1)$$

A proof of (1) proceeds by rule induction, an application of which yields 8 subgoals. The goal related to the Fake protocol rule is the following:[4]

$$\begin{array}{l}[evsf \in \mathsf{otway}\ ;\ \mathsf{Key}\ (\mathsf{shrK}\ A) \in \mathsf{parts}\ (\mathsf{knows}\ \mathsf{Spy}\ evsf) = A \in \mathsf{bad}\ ; \\ \ X \in \mathsf{synth}\ (\mathsf{analz}\ (\mathsf{knows}\ \mathsf{Spy}\ evsf))] \\ \vdash \forall B.\ \mathsf{Key}\ (\mathsf{shrK}\ A) \in \mathsf{parts}\ (\mathsf{knows}\ \mathsf{Spy}\ \boxed{(\mathsf{Says}\ \mathsf{Spy}\ BX)\#\underline{evsf}}) = A \in \mathsf{bad}\end{array}$$

Notice the similarity between this goal and the Fake inference rule of the protocol (see Fig. 2.3). A simple ripple proof, applies the rules below:[5]

$$\begin{array}{rcl}\mathsf{knows}\ \mathsf{Spy}\ (\boxed{(\mathsf{Says}\ A\ B\ X)\#\underline{evs}}) & = & \boxed{X :: \underline{\mathsf{knows}\ \mathsf{Spy}\ evs}} \\ \mathsf{parts}\ (\boxed{X :: \underline{H}}) & = & \boxed{\mathsf{parts}\ \{X\} \cup \underline{\mathsf{parts}\ H}} \\ x \in \boxed{s \cup \underline{t}} & = & \boxed{x \in s \vee \underline{x \in t}}\end{array} \qquad (2)$$

where :: denotes the set constructor function. Notice that including (2) in any term-rewriting system inevitably yields non-termination.

Fertilisation is applicable to the current goal; its application leaves us with a new goal of the form:

$$\begin{array}{l}[\mathsf{otway}\ evsf;\ \mathsf{Key}\ (\mathsf{shrK}\ A) \in \mathsf{parts}\ (\mathsf{knows}\ \mathsf{Spy}\ evsf) = A \in \mathsf{bad} \\ \ X \in \mathsf{synth}\ (\mathsf{analz}\ (\mathsf{knows}\ \mathsf{Spy}\ evsf))] \\ \vdash \forall B.\ (\mathsf{Key}\ (\mathsf{shrK}\ A) \in \mathsf{parts}\ \{X\} \vee A \in \mathsf{bad}\ ) = A \in \mathsf{bad}\end{array}$$

A further simplification, transforms this goal into a new one:

$$\begin{array}{l}[\mathsf{otway}\ evsf;\ \mathsf{Key}\ (\mathsf{shrK}\ A) \in \mathsf{parts}\ \{X\};\ X \in \mathsf{synth}\ (\mathsf{analz}\ (\mathsf{knows}\ \mathsf{Spy}\ evsf))] \\ \vdash \mathsf{Key}\ (\mathsf{shrK}\ A) \in \mathsf{parts}\ (\mathsf{knows}\ \mathsf{Spy}\ evsf)\end{array} \qquad (3)$$

[3] To obtain a HOL implementation of the inductive approach, the reader is referred to `http://webdia.cem.itesm.mx/ac/raulm/pub/33337-A`.

[4] Notice that, for the sake of simplicity, we have already included the annotations.

[5] We emphasise that wave-rules are generated dynamically, from the symbol definitional (in)equations, each of which may give rise to many different wave-rules.

Notice how rippling moves the wave-fronts outwards, up to the top of the term tree structure: the boxes are dominating the goal formula structure. When no further rippling is applicable, fertilisation can be often applied to simplify the goal. Since wave-rules must preserve the skeleton of the annotated goal, the search space induced by rippling is less than that induced by rewriting at the object-level. Thus, `difference_reduction`, inhabiting `step_case`, decreases the proof search space.

However, our experiments show that in order to establish most security goals, often we need to iterate several applications of `difference_reduction`. This is because, a single application of this method outputs a goal that is either trivially established, via `base_case` for example, or that has little structure for the ripple. Equation (3) is an example goal with no structure to be exploited by ripple. In these cases, one has to manually link the goals to enable further ripple.

Fortunately, linking hypotheses to enable proof discovery is a task that can be easily automated, by fully saturating the hypotheses, but at the expense of increasing the search space. We are currently working on a method to prescriptively drive the normalisation of a goal to an intended stage, thus keeping the search space moderately small.

Going back to our running example, since `parts` is monotonic with respect to subset, it is easy to transform the hypothesis list as follows:

$$\begin{aligned}&[\mathsf{otway}\ evsf;\ \mathsf{Key}\ (\mathsf{shrK}\ A) \in \mathsf{parts}\ \{X\};\ X \in \mathsf{synth}\ (\mathsf{analz}\ (\mathsf{knows\ Spy}\ evsf));\\ &\ \mathsf{Key}\ (\mathsf{shrK}\ A) \in \mathsf{parts}\ (\mathsf{synth}\ (\mathsf{analz}\ (\mathsf{knows\ Spy}\ evsf)))]\\ &\vdash \mathsf{Key}\ (\mathsf{shrK}\ A) \in \mathsf{parts}\ (\mathsf{knows\ Spy}\ evsf)\end{aligned}$$

This goal can be further rippled, only that the ripple would have to be performed on the hypotheses. In these cases, we use the contrapositive of an implication:

$$P \to Q \vdash \neg Q \to \neg P \tag{4}$$

before annotating the goal. Using this simple maneuver, we have avoided the need of writing new HOL tactics that could do the ripple on the hypotheses. Thus, applying (4) and introducing the annotations, we have to prove that:

$$\begin{aligned}&[\mathsf{otway}\ evsf;\ldots;\neg(\mathsf{Key}\ (\mathsf{shrK}\ A) \in \mathsf{parts}\ (\mathsf{knows\ Spy}\ evsf))]\\ &\vdash \neg(\mathsf{Key}\ (\mathsf{shrK}\ A) \in \mathsf{parts}\ (\boxed{\mathsf{synth}\ (\mathsf{analz}\ (\underline{\mathsf{knows\ Spy}\ evsf}))}))\end{aligned}$$

In this case, rippling applies the rules below:

$$\mathsf{parts}\ (\boxed{\mathsf{synth}\ \underline{H}}) = \boxed{\underline{\mathsf{parts}\ H} \cup \mathsf{synth}\ H} \qquad \mathsf{parts}\ (\boxed{\mathsf{analz}\ \underline{H}}) = \mathsf{parts}\ H$$

$$x \in \boxed{\underline{s} \cup t} = \boxed{\underline{x \in s} \lor x \in t} \qquad \neg(\boxed{\underline{A} \lor B}) = \boxed{(\underline{\neg A} \land \neg B)}$$

This leaves a subgoal that it is trivially established, using the additional results:

$$\begin{aligned} \mathsf{Key}\ K \in \mathsf{synth}\ H &= \mathsf{Key}\ K \in H \\ X \in \mathsf{analz}\ H &\to X \in \mathsf{parts}\ H \end{aligned}$$

Table 1 shows a few security goals, all related to the Otway-Rees protocol, in which `difference_reduction` was used to drive the search for a proof. Space constraints prevent us from presenting a complete proof. Yet, we have found out that the above proof pattern appears often in a number of proofs of security goals. The full test set, including HOL, HOL-Clam, Clam and the methods for protocol verification, are available upon request, by sending electronic-mail to the first author.

Proof planning offers a number of techniques to automate theorem proving. While rippling is at the heart of the inductive proof plan, there are other techniques that are worth mentioning. For instance, the `base_case` method makes use of an recursive-path ordering (RPO) in order to fully automatically extract rewrite rules from symbol definitions. This RPO term-rewriting system is not proof equivalent to Isabelle's simplifier. This is due to the fact that `base_case` includes rewrite rules obtained from inequalities and implications. By comparison, within Isabelle, implication properties are normally used for elimination or destruction, never for simplification.

Our method can partially drive the search for a proof of a security goal. By contrast, the Isabelle blast method is able to proof a number of goals without interaction. This is because it saturates the hypotheses in a way that most of their logical consequences are pulled out. When an single application of blast

Table 1. Otway-Rees security goals solved using difference reduction

No.	Property
1	$evs \in \mathsf{otway} \to \mathsf{Gets}\ B\ X \in \mathsf{set}\ evs \to \exists A.\ \mathsf{Says}\ A\ B\ X \in \mathsf{set}\ evs$
2	$evs \in \mathsf{otway} \to \neg(A \in bad) \to$ $\mathsf{Crypt}\ (\mathsf{shrK}\ A)(\{\!|N_A, \mathsf{Agent}\ A, \mathsf{Agent}\ B|\!\}) \in \mathsf{parts}\ (\mathsf{knows}\ \mathsf{Spy}\ evs) \to$ $\neg(\mathsf{Crypt}\ (\mathsf{shrK}\ A)(\{\!|N'_A, N_A, \mathsf{Agent}\ A', \mathsf{Agent}\ A|\!\}) \in \mathsf{parts}\ (\mathsf{knows}\ \mathsf{Spy}\ evs))$
3	$evs \in \mathsf{otway} \to \neg(A \in \mathsf{bad}\ ) \to$ $\mathsf{Crypt}\ (\mathsf{shrK}\ A)(\{\!|N_A, \mathsf{Agent}\ A, \mathsf{Agent}\ B|\!\}) \in \mathsf{parts}\ (\mathsf{knows}\ \mathsf{Spy}\ evs) \to$ $\mathsf{Says}\ A\ B$ $\{\!|N_A, \mathsf{Agent}\ A, \mathsf{Agent}\ B, \{\!|N_A, \mathsf{Agent}\ A, \mathsf{Agent}\ B|\!\}_{K_{AS}}|\!\} \in \mathsf{set}\ evs$
4	$\neg(A \in \mathsf{bad}\ ) \to \mathsf{otway}\ evs \to$ $\mathsf{Crypt}\ (\mathsf{shrK}\ A)(\{\!|N_A, \mathsf{Agent}\ A, \mathsf{Agent}\ B|\!\}) \in \mathsf{parts}\ (\mathsf{knows}\ \mathsf{Spy}\ evs) \to$ $\mathsf{Crypt}\ (\mathsf{shrK}\ A)(\{\!|N_A, \mathsf{Agent}\ A, \mathsf{Agent}\ C|\!\}) \in \mathsf{parts}\ (\mathsf{knows}\ \mathsf{Spy}\ evs) \to$ $(B = C)$
5	$\neg(A \in \mathsf{bad}\ ) \to evs \in \mathsf{otway} \to$ $\mathsf{Says}\ A\ B\{\!|N_A, \mathsf{Agent}\ A, \mathsf{Agent}\ B, \{\!|N_A, \mathsf{Agent}\ A, \mathsf{Agent}\ B|\!\}_{K_{AS}}|\!\} \in \mathsf{set}\ evs \to$ $\{\!|N_A, (\mathsf{Key}\ K)|\!\} \in \mathsf{parts}\ (\mathsf{knows}\ \mathsf{Spy}\ evs) \to$ $(\exists N_B.; \mathsf{Says}\ \mathsf{S}\ B$ $\{\!|N_A, \{\!|N_A, (\mathsf{Key}\ K)|\!\}_{K_{AS}}, \{\!|N_B, (\mathsf{Key}\ K)|\!\}_{K_{BS}}|\!\} \in \mathsf{set}\ evs)$

fails, however, one needs to be an expert to find the required switch or identify the property that is still missing for a proof to be found.

5 Further Work and Conclusions

Further work involves searching for an algorithm which automatically and prescriptively could identify how to link several hypotheses to allow more ripple. This would enable us to proof simple security goals completely. Further work also involves formulating a proof method to automatically guide the application of rule induction. This method would play a key role in a proof plan for rule induction.

Our results are encouraging. We believe proof planning can help structuring proofs of security protocols so as to reduce both the human skill levels and the development time required to verify an security protocol. Moreover, we believe proof plan may help understand how to use failure in the search for a proof so as to either suggest high-level, intelligible changes to the structure of a faulty protocol, or synthesise an attack.

References

1. Paulson, L.C.: The Inductive Approach to Verifying Cryptographic Protocols. Journal of Computer Security **6** (1998) 85–128
2. Bundy, A.: The Use of Explicit Plans to Guide Inductive Proofs. In Lusk, R., Overbeek, R., eds.: Proceedings of the 9th Conference on Automated Deduction. Lecture Notes in Computer Science, Vol. 310, Argonne, Illinois, USA, Springer-Verlag (1988) 111–120x
3. Bundy, A., van Harmelen, F., Hesketh, J., Smaill, A.: Experiments with proof plans for induction. Journal of Automated Reasoning **7** (1991) 303–324
4. Basin, D., Walsh, T.: Difference unification. In Bajcsy, R., ed.: Proceedings of the 13th International Joint Conference on Artificial Intelligence, IJCAI'93. Volume 1., San Mateo, CA, Morgan Kaufmann (1993) 116–22
5. Bundy, A., Stevens, A., van Harmelen, F., Ireland, A., Smaill, A.: Rippling: A heuristic for guiding inductive proofs. Artificial Intelligence **62** (1993) 185–253
6. Boulton, R., Slind, K., Bundy, A., Gordon, M.: An interface between CLAM and HOL. In Grundy, J., Newey, M., eds.: 11th International Conference on Theorem Proving in Higher-Order Logics (TPHOLs'98), Camberra, Australia, Springer-Verlag (1998) 87–104 Lecture Notes in Computer Science, Vol. 1479.
7. Gordon, M.: HOL: A proof generating system for higher-order logic. In Birtwistle, G., Subrahmanyam, P.A., eds.: VLSI Specification, Verification and Synthesis, Kluwer (1988)
8. Bundy, A., van Harmelen, F., Horn, C., Smaill, A.: The Oyster-Clam system. In Stickel, M.E., ed.: Proceedings of the 10th International Conference on Automated Deduction. Lecture Notes in Artificial Intelligence, Vol. 449, Springer-Verlag (1990) 647–648
9. Monroy, R., Bundy, A., Green, I.: Planning Proofs of Equations in CCS. Automated Software Engineering **7** (2000) 263–304

10. Schneider, S.: Modelling Security Properties with CSP. Computer Science Department, Technical Report Series CSD-TR-96-04, Royal Holloway, University of London (1996)
11. Paulson, L.C.: Isabelle: the next 700 theorem provers. In Odifreddi, P., ed.: Logic and Computer Science, Academic Press (1990) 77–90

On Some Differences Between Semantics of Logic Program Updates

João Alexandre Leite

CENTRIA, New University of Lisbon, Portugal

Abstract. Since the introduction of logic program updates based on causal rejection of rules, several different semantics were set forth. All these semantics were introduced with the same underlying motivation i.e., to overcome the drawbacks of interpretation based updates by applying the principle of inertia to program rules, but they were all defined for different classes of logic programs thus making their comparisons difficult. In this paper we redefine such existing semantics, and set forth a new one, all in the more general setting of Generalized Logic Programs, in a way that facilitates their comparisons. Subsequently, we take a closer look at the subtle differences between these otherwise similar approaches.

1 Introduction

Concerning modifications to a knowledge base represented by a propositional theory, two abstract frameworks have been distinguished in [16] and [7]. One, theory revision, deals with incorporating new knowledge about a static world whose previous representation was incomplete or incorrect. The other deals with changing worlds, and is known as theory update. This paper addresses the issue of updates of logic programs, a subject that was recently put into context in [5].

Until recently, the work devoted to this issue followed the so called interpretation update approach based on the idea of reducing the problem of finding the update of a knowledge base by another knowledge base to the problem of finding the updates of its individual models. In [13] the authors introduce the framework of *Revision Programming*[1] where they allow the specification of updates of interpretations.

In [9], it is pointed out that such approach suffers from important drawbacks when the initial knowledge base is not purely extensional (it contains rules), and propose a new approach where the principle of inertia is applied to the rules of the initial knowledge base, rather than to its model literals. This led to the paradigm of *Dynamic Logic Programming*. A *Dynamic Logic Program* (DLP) is a sequence of Logic Programs where each represents a time period (state) and contains some knowledge that is supposed to be true at the state. The mutual relationships existing between different states (specified as the ordering relation) are then used to determine its declarative semantics.

Since the introduction of this form of updates of logic programs, several different semantics were set forth [9,10,2,4,1,8], with the motivation of overcoming the drawbacks of interpretation based updates by applying the principle of inertia to program

[1] Despite the name, the authors are actually dealing with updates and not revisions.

rules. But they were all defined for different classes of logic programs, thus making their comparisons difficult. For example, [9,10] define the semantics of *Justified Updates* for sequences of Revision Programs (a variant of Logic Programs), [2,8] define the semantics of *Stable Models* for sequences of *Generalized (Extended) Logic Programs (GLP)* (Logic Programs allowing both default and strong negation in the heads of rules), and [4] define the semantics of Update Answer Sets for sequences of Extended Logic Programs. In [4], bridges between these semantics are established for restricted classes of logic programs, showing that under some restrictions the semantics coincide.

In this paper we take a closer look at the differences between these similar approaches. For this, we start by establishing the definitions of six different semantics, all set forth in a similar manner, thus making their comparisons easier. Four of these six semantics, all defined for sequences of *GLP* [2], either coincide or generalize the semantics mentioned before. The remaining two are new proposals. Subsequently, we compare all six semantics, either by means of examples or by means of properties, mostly with the intention of bringing out their differences rather than their similarities.

The paper is structured as follows: in Sect. 2 we recall some definitions; in Sect 3 we define the six mentioned semantics and relate them to the ones in the literature; in Sect 4 we establish some comparisons; in Sect 5 we wrap up.

2 Preliminaries

Let $\mathcal{A}$ be a set of propositional atoms. An **objective literal** is either an atom A or a strongly negated atom $\neg A$. A **default literal** is an objective literal preceded by *not*. A **literal** is either an objective literal or a default literal. A **rule** r is an ordered pair $H(r) \leftarrow B(r)$ where $H(r)$ (dubbed the head of the rule) is a literal and $B(r)$ (dubbed the body of the rule) is a finite set of literals. A rule with $H(r) = L_0$ and $B(r) = \{L_1, \ldots, L_n\}$ will simply be written as $L_0 \leftarrow L_1, \ldots, L_n$. A **tautology** is a rule of the form $L \leftarrow Body$ with $L \in Body$. A **generalized logic program** (*GLP*) *P*, in $\mathcal{A}$, is a finite or infinite set of rules. A program is called an **extended logic program** *(ELP)* if no default literals appear in the heads of its rules. If $H(r) = A$ (resp. $H(r) = not\, A$) then $not\, H(r) = not\, A$ (resp. $not\, H(r) = A$). If $H(r) = \neg A$, then $\neg H(r) = A$. By the **expanded generalized logic program** corresponding to the GLP *P*, denoted by $\mathbf{P}$, we mean the GLP obtained by augmenting P with a rule of the form $not\, \neg H(r) \leftarrow B(r)$ for every rule, in *P*, of the form $H(r) \leftarrow B(r)$, where $H(r)$ is an objective literal. An **interpretation** M of $\mathcal{A}$ is a set of objective literals that is consistent i.e., M does not contain both A and $\neg A$. An objective literal L is true in M, denoted by $M \vDash L$, iff $L \in M$, and false otherwise. A default literal $not\, L$ is true in M, denoted by $M \vDash not\, L$, iff $L \notin M$, and false otherwise. A set of literals B is true in M, denoted by $M \vDash B$, iff each literal in B is true in M. An interpretation M of $\mathcal{A}$ is an **answer set** of a GLP P iff $M' = least\,(\mathbf{P} \cup \{not\, A \mid A \notin M\})$, where $M' = M \cup \{not\_A \mid A \notin M\}$, A is an objective literal, and *least*(.) denotes the least model of the definite program

[2] For motivation on the need for GLPs in updates, the reader is referred to [2,8]. Note that within the context of updates, GLPs are not equivalent to Logic Programs with Integrity Constraints.

obtained from the argument program by replacing every default literal *not A* by a new atom *not_A*. Let *AS (P)* denote the set of answer-sets of *P*.

A **dynamic logic program** *(DLP)* is a sequence of generalized logic programs. Let $\mathcal{P} = (P_1, ..., P_s)$, $\mathcal{P}' = (P_1', ..., P_n')$ and $\mathcal{P}'' = (P_1'', ..., P_s'')$ be DLPs. We use $\rho(\mathcal{P})$ to denote the multiset of all rules appearing in the programs $\mathbf{P}_1, ..., \mathbf{P}_s$, and $(\mathcal{P}, \mathcal{P}')$ to denote $(P_1, ..., P_s, P_1', ..., P_n')$ and $\mathcal{P} \cup \mathcal{P}''$ to denote $(P_1 \cup P_1'', ..., P_s \cup P_s'')$.

3 Semantics of Updates

In this Section we set forth the definitions of several semantics for dynamic logic programs in a uniform manner. Subsequently we establish that some of these semantics either coincide or generalize the semantics proposed in the literature. Without loss of generality, we only consider the semantics at the last state of the DLP. The common aspect that relates the semantics for updates addressed here, known as those *based on causal rejection ofrules,* is their relying on the notion that a newer rule that is in conflict with an older one may reject it to avoid contradiction. We start by defining the notion of conflicting rules as follows: two rules r and r' are **conflicting,** denoted by $r \bowtie r'$, iff $H(r) = not\, H(r')$. Note that we do not establish a pair of rules whose heads are the strong negation of one another as being conflicting. Intuitively, these rules should be regarded as conflicting. They are not explicitly stated as such since, by using the expanded versions of GLPs, we have that for every pair of rules r_1 and r_2 in a DLP such that $H(r_1) = \neg H(r_2)$ there are two rules r_1' and r_2', introduced by the expansion operation, that are conflicting with the original ones i.e. $r_1 \bowtie r_2'$ and $r_1' \bowtie r_2$. As will become clear below, this is enough to accomplish the desired rejection when a newer rule, whose head is the strongly negated atom of the rule being rejected, exists.

Next we define three notions ofrejected rules in a DLP. Intuitively, when we consider an interpretation, a rule that belongs to a program of a DLP should be rejected if there is a newer conflicting rule whose body is true in the interpretation. This amounts to one notion of rejection. The second builds upon the first and allows for rules of the same state to reject each other. Intuitively this may seem odd but, as will be seen below, it produces desirable results. The third notion builds upon the first to further impose that the rejector rule is itself not rejected. Intuitively this condition seems reasonable but, as will be seen below, it produces results that may not be desirable.

Definition 1 (Rejected Rules). *Let* $\mathcal{P} = (P_1, ..., P_s)$ *be a* DLP *and M an interpretation. We define:*

$$Rej(\mathcal{P}, M) = \{r \mid r \in \mathbf{P}_i, \exists r' \in \mathbf{P}_j, i < j, r \bowtie r', M \models B(r')\}$$
$$Rej^{+}(\mathcal{P}, M) = \{r \mid r \in \mathbf{P}_i, \exists r' \in \mathbf{P}_j, i \leq j, r \bowtie r', M \models B(r')\}$$
$$Rej^{*}(\mathcal{P}, M) = \{r \mid r \in \mathbf{P}_i, \exists r' \in \mathbf{P}_j \setminus Rej^{*}(\mathcal{P}, M), i < j, r \bowtie r', M \models B(r')\}$$

Before we define the semantics, we turn our attention to the notion of default assumptions. In Section 2, when we defined the semantics of answer-sets for GLPs, we purposely did it in a somehow non standard way. Instead of using the standard GL transformation [6] (or, to be more precise, its modified version for GLPs [12]), we explicitly add the default assumptions (default literals for all those objective literals that do not belong to the considered interpretation) to the program and then determine its least model. Similarly, when determining the semantics for DLPs, one also considers the least model of a program, consisting of all rules belonging to the programs of the DLP, without the rejected rules, together with a set of default assumptions. But unlike for the answer-set semantics above, not all the semantics for DLPs will allow for the 7explicit addition of *every* default literal whose corresponding objective literal does not belong to the interpretation being considered. Instead, for such DLP semantics, the default assumptions *(not A)* are restricted to those corresponding to unsupported objective literals i.e., those objective literals *A* for which there is no rule in the DLP whose body is true in the interpretation being considered. The intuition behind this is that if there exists a rule in the DLP (rejected or not) that would support the truth of some objective literal *A,* then *A* should not be able to be assumed*false by default.* Instead, its falsity, to exist, should only be obtained by some newer rule r that forces it to be false i.e., one with $H(r) = not\ A$ and $M \models B(r)$ (where *M* is the interpretation being considered). Otherwise, we can have situations where the assumption that some objective literal *A* is false indirectly leads to the rejection of a fact *A,* as will be shown below. Since not all semantics impose this restriction, we define two notions of default assumptions:

Definition 2 (Default Assumptions). *Let* $\mathcal{P} = (P_1, ..., P_s)$ *be a* DLP *and M an interpretation. Define (where A is an objective literal):* $Def^*(\mathcal{P}, M) = \{not\ A \mid A \notin M\}$ *and* $Def(\mathcal{P}, M) = \{not\ A \mid \nexists r \in \rho(\mathcal{P}), H(r) = A, M \models B(r)\}$.

Using each combination of rejected rules and default assumptions, we are now ready to define six distinct semantics for DLPs, all of which based on the intuition that some interpretation is a model according to a semantics iff it obeys an equation based on the least model of the multiset of all the rules in the (expanded) DLP, without those rejected rules, together with a set of default assumptions. The six semantics are dubbed *dynamic stable model semantics (DSM), dynamic justified update semantics (DJU), dynamic u-model semantics (DUM), dynamic answer-set semantics (DAS), refined dynamic stable model semantics (RDSM)* and *refined dynamic justified update semantics (RDJU).*

Definition 3 (Semantics of Updates). *Let* $\mathcal{P} = (P_1, ..., P_s)$ *be a* DLP *and M an interpretation. Then:*

DSM: *M is a* dynamic stable model *of* $\mathcal{P}$ *iff*
$$M' = least(\rho(\mathcal{P}) - Rej(\mathcal{P}, M) \cup Def(\mathcal{P}, M))$$
DJU: *M is a* dynamic justified update *of* $\mathcal{P}$ *iff*
$$M' = least(\rho(\mathcal{P}) - Rej(\mathcal{P}, M) \cup Def^*(\mathcal{P}, M))$$
DUM: *M is a* dynamic u-model *of* $\mathcal{P}$ *iff*
$$M' = least(\rho(\mathcal{P}) - Rej^*(\mathcal{P}, M) \cup Def(\mathcal{P}, M))$$
DAS: *M is a* dynamic answer-set *of* $\mathcal{P}$ *iff*
$$M' = least(\rho(\mathcal{P}) - Rej^*(\mathcal{P}, M) \cup Def^*(\mathcal{P}, M))$$
RDSM: *M is a* refined dynamic stable model *of* $\mathcal{P}$ *iff*
$$M' = least(\rho(\mathcal{P}) - Rej^+(\mathcal{P}, M) \cup Def(\mathcal{P}, M))$$
RDJU: *M is a* refined dynamic justified update *of* $\mathcal{P}$ *iff*
$$M' = least(\rho(\mathcal{P}) - Rej^+(\mathcal{P}, M) \cup Def^*(\mathcal{P}, M))$$

where M', $\rho(.)$*and* $least(.)$ *are as before. Let* $DSM(\mathcal{P})$*denote the set ofall* dynamic stable models *of* $\mathcal{P}$, $DJU(\mathcal{P})$*the set ofall* dynamic justified updates *of* $\mathcal{P}$, $DUM(\mathcal{P})$

the set of all dynamic u-models *of* $\mathcal{P}$, $DAS(\mathcal{P})$ *the set of all* dynamic answer-sets *of* $\mathcal{P}$, $RDSM(\mathcal{P})$ *the set of all* refined dynamic stable models *of* $\mathcal{P}$, *and* $RDJU(\mathcal{P})$ *the set of all* refined dynamic justified updates *of* $\mathcal{P}$.

We now relate these semantics with those defined in the literature.

Proposition 1. *Let* $\mathcal{P}$ *be a* DLP. *M is a* dynamic stable model *of* $\mathcal{P}$ *iff M is a* stable model *of* $\mathcal{P}$ *(as of [2,8]).*

Remark 1. Strictly speaking, if we consider the definitions in [2], the result only holds for normal generalized logic programs (i.e. without strong negation). This is so because of an incorrection in the original definition which has been corrected in [8].

Proposition 2. *Let* $\mathcal{P}$ *be a* DLP. *M is a* dynamic justified update *of* $\mathcal{P}$ *iff M is a* $\mathcal{P}$**−justified** update *(as of [10,9]).*

Remark 2. The semantics of $\mathcal{P}$***−justified** updates* was originally established in a setting where *Revision Programs* [13] were used instead of GLP's. We consider the trivial correspondence between GLPs and Revision Programs where each *in* (A) (resp.*out* (A)) of the latter corresponds to a A (resp *not* A) of the former.

The *dynamic answer-set semantics* generalizes the *update answer-set semantics* [4], originally defined for DLPs consisting of extended logic programs only i.e., without default negation in the heads of rules, to the case of DLPs consisting of GLPs.

Proposition 3. *Let* $\mathcal{P} = (P_1, ..., P_s)$ *be a* DLP *where each* P_i *is an extended logic program. M is a* dynamic answer-set *of* $\mathcal{P}$ *iff M is an* update answer-set *(as of [4]).*

In [1] the use of strong negation is not allowed. The *refined dynamic stable model semantics* defined here extends the semantics of [1] to allow for strong negation.

Proposition 4. *Let* $\mathcal{P} = (P_1, ..., P_s)$ *be a* DLP *where each* P_i *is a normal generalized logic programs (i.e. without strong negation). M is a* refined dynamic stable model *of* $\mathcal{P}$ *iff M is a* refined dynamic stable model *(as of [1]).*

The *dynamic u-model semantics* and the *refined dynamic justified update semantics* do not correspond to any previously defined semantics. Their definitions are only justified for the sake of completeness since they both suffer from some drawbacks.

4 Properties and Comparisons

In this Section we compare all six semantics by means of some properties and examples that illustrate their similarities and differences. We start by relating all semantics with the answer-set semantics:

Proposition 5 (Generalization of Answer-Set Semantics). *Let* $\mathcal{P} = (P)$ *be a DLP consisting of a single GLP. Then* $DSM(\mathcal{P}) = DJU(\mathcal{P}) = DUM(\mathcal{P}) = DAS(\mathcal{P}) = RDSM(\mathcal{P}) = AS(P)$

A similar proposition does not hold for the *refined dynamic justified update semantics* ($P = \{a \leftarrow; \, not\, a \leftarrow\}$ serves as a counter-example). Since this semantics fails to obey this simple and desirable property, we will not consider it further in this paper. Next we relate the sets of models that characterize each semantics:

Theorem 1. *Let $\mathcal{P} = (P_1, ..., P_s)$ be a* DLP. *Then $RDSM(\mathcal{P}) \subseteq DSM(\mathcal{P}) \subseteq DJU(\mathcal{P}) \subseteq DAS(\mathcal{P})$ and $RDSM(\mathcal{P}) \subseteq DSM(\mathcal{P}) \subseteq DUM(\mathcal{P}) \subseteq DAS(\mathcal{P})$.*

Example 1. Let $\mathcal{P} = (P_1, P_2, P_3)$ be the *DLP* where $P_1 = \{a \leftarrow\}$, $P_2 = \{not\, a \leftarrow\}$, and $P_3 = \{a \leftarrow a\}$. We obtain $RDSM(\mathcal{P}) = DSM(\mathcal{P}) = DJU(\mathcal{P}) = \{\{\}\}$ and $DUM(\mathcal{P}) = DAS(\mathcal{P}) = \{\{\}, \{a\}\}$.

This example serves to illustrate the difference in the results caused by allowing rejected rules to reject rules or otherwise. For those semantics that use $Rej(\mathcal{P}, M)$ and $Rej^+(\mathcal{P}, M)$ (RDSM, DSM and DJU), we only obtain one model, namely $\{\}$. Since, with $Rej(\mathcal{P}, M)$ and $Rej^+(\mathcal{P}, M)$, rejected rules can reject other rules, the rule $a \leftarrow$ in P_1 is always rejected by the rule $not\ a \leftarrow$ in P_2, independently of the interpretation M being considered. Then, since there are no rules that support a (the rule $a \leftarrow a$ in P_3 is not sufficient, by itself), we cannot have a model such that a belongs to it. The interpretation $\{\}$ is then the only one that verifies the condition to be a model. Note that this is valid for RDSM, DSM and DJU. For those semantics that use $Rej^*(\mathcal{P}, M)$ (DUM and DAS), since rejected rules are not allowed to reject other rules, the rejection of the rule $a \leftarrow$ in P_1 now depends on the rule $not\, a \leftarrow$ in P_2 being rejected or not. If we consider the interpretation $\{\}$, then the rule $a \leftarrow$ is rejected and $\{\}$ verifies the condition to be a model. If we consider the interpretation $\{a\}$, then the rule $a \leftarrow a$ in P_3 rejects the rule $not\, a \leftarrow$ in P_2 and, consequently, $a \leftarrow$ in P_1 is no longer rejected. Since $least(a \leftarrow; a \leftarrow a) = \{a\}$, we have that $\{a\}$ is also a model according to DUM and DAS. By inspecting $\mathcal{P}$, we argue that the intuitive result should be to allow for one model only, namely $\{\}$. Since we are dealing with updates, the rule $not\, a \leftarrow$ in P_2 should be understood as a change in the world into a state where a is unconditionally not true. Therefore, we should not be able to reuse the rule $a \leftarrow$ in P_1 again. According to DUM and DAS, this old rule in P_1 serves as the support for itself not to be rejected i.e. it serves as the support for a, rendering the rule $a \leftarrow a$ in P_3 one that rejects $not\, a \leftarrow$ in P_2, and therefore not able to reject $a \leftarrow$ in P_1. The reader may find this line of argumentation more convincing after replacing the proposition a with the proposition *alive*. Below, we come back to this issue when we define a property that encodes the intuition behind it, which holds for RDSM, DSM and DJU, but not for DUM and DAS.

Example 2. Consider $\mathcal{P} = (P_1, P_2)$ to be the *DLP* where $P_1 = \{a \leftarrow\}$ and $P_2 = \{not\, a \leftarrow not\, a\}$. We obtain $RDSM(\mathcal{P}) = DSM(\mathcal{P}) = DUM(\mathcal{P}) = \{\{a\}\}$ and $DJU(\mathcal{P}) = DAS(\mathcal{P}) = \{\{\}, \{a\}\}$.

With this example we can observe the different results obtained by using either $Def(\mathcal{P}, M)$ or $Def^*(\mathcal{P}, M)$. When $Def(\mathcal{P}, M)$ is used (RDSM, DSM and DUM), then only one model is obtained namely $\{a\}$. If we use $Def^*(\mathcal{P}, M)$ (DJU and DAS), the model $\{\}$ also exists. This model is obtained by assuming a to be false by default i.e. $not\ A \in Def^*(\mathcal{P}, M)$. This default assumption justifies the use of the rule $not\ a \leftarrow not\ a$ in P_2 to reject the fact $a \leftarrow$ in P_1. Arguably, if we look at the rule $not\ a \leftarrow not\ a$ as a tautological one, it is fair to expect it not to be responsible, by itself, for a change in the semantics. We argue that this DLP should only have one model, namely $\{a\}$.

The previous example leads to the next property, which serves as argument in favor of using $Def\,(\mathcal{P}, M)$ instead of $Def^*\,(\mathcal{P}, M)$. It relates these semantics with *Revision Programming* [13]. Such framework, which for lack of space we cannot formally recapitulate here, characterizes the interpretations that should be accepted as the result of the update of an initial interpretation by a revision program. Such accepted interpretations are called M-justified updates. If we encode the initial interpretation as a purely extensional GLP and make it the first program of a DLP with two programs, where the second encodes the revision program, then it is desirable that a semantics for the DLP coincide with the one provided by the interpretation updates. It turns out that only the three semantics that use $Def\,(\mathcal{P}, M)$ coincide with interpretation updates.

Definition 4 (Generalization of Interpretation Updates). *Let M be an interpretation and RP a revision program (according to [13]). Let P_{RP} be the GLP obtained from RP by replacing atoms of the form in (L) (resp. out (L)) with L (resp not L). Let $P_M = \{A \leftarrow \mid A \in M\}$. We say that an update semantics SEM generalizes Interpretation Updates (IU)(in the sense of[13]) iff for every M and P_{RP} it holds that an interpretation I is a M-justified update iff it is a model of (P_M, P_{RP}) according to SEM i.e., $I \in SEM\,((P_M, P_{RP}))$.*

Theorem 2 (Generalization of Interpretation Updates). *RDSM, DSM and DUM generalize IU. DJU and DAS do not generalize IU.*

The DLP in Example 2 serves as a counter example to show that DJU and DAS do not generalize *IU* because according to the latter, $\{a\}$ is the only M-justified update.

From the exposition so far, it becomes apparent that the differences between the semantics are very subtle, and all related to the extent that these semantics are immune to tautologies. Immunity to tautologies is desirable and can be defined as follows:

Definition 5 (Immunity to Tautologies). *An update semantics SEM is immune to tautologies iff for any DLP $\mathcal{P} = (P_1, ..., P_s)$ and any sequence of sets of tautologies $\mathcal{E} = (E_1, ..., E_s)$, it holds that $SEM\,(\mathcal{P}) = SEM\,(\mathcal{P} \cup \mathcal{E})$.*

Theorem 3. *DSM, DUM, DJU and DAS are not immune to tautologies.*

Example 3. Consider the *DLP* $\mathcal{P} = (P_1, P_2)$ where $P_1 = \{a \leftarrow; not\,a \leftarrow\}$ and $P_2 = \{a \leftarrow a\}$. All *DSM, DUM, DJU* and *DAS* have a single model, namely $\{a\}$. According to these four semantics, if one program is contradictory, a tautological update has the effect of removing such contradiction.

Theorem 4. *[1] RDSM is immune to tautologies.*

Before we proceed, we establish a notion of equivalence between two DLPs under some update semantics. Intuitively two DLPs are equivalent if they have the same semantics and, when both are updated by the same arbitrary sequence of programs (i.e. the updating sequence is appended to both DLPs), their semantics still coincides.

Definition 6 (Update Equivalence). *Two DLPs $\mathcal{P}_\alpha$ and $\mathcal{P}_\beta$ are* update equivalent *under semantics SEM, denoted by $\mathcal{P}_\alpha \overset{SEM}{\equiv} \mathcal{P}_\beta$, iff for every DLP $\mathcal{P}$, it holds that $SEM((\mathcal{P}_\alpha,\mathcal{P})) = SEM((\mathcal{P}_\beta,\mathcal{P}))$.*

This notion of equivalence is important inasmuch as it allows us to replace a DLP that describes the history of some world by a simpler one, if we are not concerned with the past history, but we want to guarantee that the present and future are preserved. Ideally, we would like to devise, for each semantics, an operator that would condense any two consecutive programs belonging to a DLP into a single one, written in the same language. By repeatedly applying such an operator we would reduce a DLP to a single GLP. Such operators have the following formal definition:

Definition 7 (General State Condensing Operator). *Let $\mathcal{A}$ be a set of propositional atoms. Let Π denote the set of all generalized logic programs over the set of atoms $\mathcal{A}$. Let Θ be an operator with signature $\Theta : \Pi \times \Pi \to \Pi$. We say that Θ is a* general state condensing operator *for language $\mathcal{A}$ and semantics SEM iff for every DLP $\mathcal{P} = (P_1, ..., P_s)$ over $\mathcal{A}$ it holds that $\mathcal{P} \overset{SEM}{\equiv} (P_1, ..., P_{i-1}, \Theta(P_i, P_{i+1}), P_{i+2}, ..., P_s)$.*

Theorem 5. *Let $\mathcal{A}$ be a non-empty language. General state condensing operators for language $\mathcal{A}$ and semantics RDSM, DSM, DJU, DUM, and DAS,* do not *exist.*

Example 4. Let $\mathcal{P} = (P_1, P_2)$ be the *DLP*, over $\mathcal{A} = \{a, b\}$, where $P_1 = \{a \leftarrow b; b \leftarrow\}$ and $P_2 = \{not\, b \leftarrow not\, a\}$. We have $RDSM(\mathcal{P}) = DSM(\mathcal{P}) = DJU(\mathcal{P}) = DUM(\mathcal{P}) = DAS(\mathcal{P}) = \{\{\}, \{a, b\}\}$. Since a general state condensing operator Θ would condense P_1 and P_2 into a single program $\Theta(P_1, P_2)$, and all five update semantics coincide with the answer-set semantics, for DLPs with a single program, then, $AS(\Theta(P_1, P_2))$ would have to be equal to $\{\{\}, \{a, b\}\}$. But there is no generalized logic program whose answer sets are $\{\}$ and $\{a, b\}$, because $\{\} \subset \{a, b\}$ and it is known that answer-sets are minimal. Similar examples written in a language containing just one propositional atom exist, from which we prove the theorem.

The definition for general state condensing operators requires the language in which the resulting program is written to be the same as that of the original DLP. If we allow extensions to the original language, then there exists, for each of the semantics introduced, a polynomial transformation that takes a DLP and produces a single GLP, written in an extended language, whose answer-sets (when restricted to the initial language) coincide with the models of the update semantics. The existence of these transformations allows for the use of available answer-set software (e.g. *SMODELS* [14] *and DLV* [11]) to determine the update semantics, by means of a preprocessor that implements the transformation. Some of these implementations are publicly available. We do not present such transformations here for lack of space (some are in the cited literature), but their existence suffices to establish the following complexity results for all the semantics (inherited from those of Logic Programming under the Answer-set semantics):

Theorem 6 (Computational Complexity). *Let $\mathcal{P}$ be a DLP. Deciding if $DSM(\mathcal{P})$ (resp. $DJU(\mathcal{P})$, $DUM(\mathcal{P})$, $DAS(\mathcal{P})$, $RDSM(\mathcal{P})$) is not empty is NP — complete. Deciding if an interpretation belongs to $DSM(\mathcal{P})$ (resp. $RDSM(\mathcal{P})$, $DUM(\mathcal{P})$,*

$DAS(\mathcal{P})$, $DJU(\mathcal{P})$) is P. Deciding if an atom is true in at least one interpretation that belongs to $DSM(\mathcal{P})$ (resp. $DJU(\mathcal{P})$, $DUM(\mathcal{P})$, $DAS(\mathcal{P})$, $RDSM(\mathcal{P})$) is NP – complete; Deciding if an atom is true in all interpretations that belong to $DSM(\mathcal{P})$ (resp. $DJU(\mathcal{P})$, $DUM(\mathcal{P})$, $DAS(\mathcal{P})$, $RDSM(\mathcal{P})$) is coNP-complete.

Since the size of the program obtained by the mentioned transformations depends on the number of rules and the number of states of the DLP, we now address ways of simplifying a DLP both by eliminating certain states and certain rules, while preserving update equivalence.

Definition 8 (State Elimination). *Let $\mathcal{P} = (P_1, ..., P_s)$ be a* DLP. *Define:*
SEa: If $P_i = \{\}$, then $\mathcal{P} \overset{SEM}{\equiv} (P_1, ..., P_{i-1}, P_{i+1}, ..., P_s)$;
SEb: If $\forall r \in P_i, r' \in P_{i+1} : r \not\bowtie r'$, then $\mathcal{P} \overset{SEM}{\equiv} (P_1, ..., P_{i-1}, P_i \cup P_{i+1}, P_{i+2}, ..., P_s)$.

The first one, *SEa,* encodes that updates by empty programs should not change the semantics and should be allowed to be removed. *SEb* encodes that two consecutive programs that have no conflicting rules should be able to be merged into a single one. We now proceed to state simplification i.e. (syntactical) removal of superfluous rules:

Definition 9 (State Simplification). *Let $\mathcal{P} = (P_1, ..., P_s)$ be a* DLP. *Define:*
SSa: If $r \in P_i$ and $\exists r' \in P_j, i < j, H(r') = H(r)$ and $B(r') \subseteq B(r)$, then $\mathcal{P} \overset{SEM}{\equiv} (P_1, ..., P_i \backslash \{r\}, ..., P_s)$;
SSb: If $r \in P_i$ and $\exists r' \in P_j, i < j, H(r') = not\, H(r)$ and $B(r') \subseteq B(r)$, then $\mathcal{P} \overset{SEM}{\equiv} (P_1, ..., P_i \backslash \{r\}, ..., P_s)$.

The first one, *SSa,* encodes that one should be able to remove an older rule for some literal, if a newer rule for that literal exists and is equal or more general (i.e. its body is a subset of the older rule's body). *SSb* encodes that one should be able to remove an older rule for some literal if a newer rule for its default complement exists and its body is equal or a subset of the older rule's body. *SSb* seems intuitive because if the body of the newer rule is always true when the body of the older rule also is, then the conclusion of the newer rule should always prevail over the conclusion of the older one.

Theorem 7. *SEa and SEb hold for RDSM, DSM, DJU, DUM and DAS. SSa holds for RDSM, DSM, DJU, DUM and DAS. SSb holdsfor RDSM, DSM and DJU. SSb does not hold for DUM and DAS.*

The following example, illustrates why *SSb* does not hold for DUM and DAS, which is related to their use of $Rej^*(\mathcal{P}, M)$, instead of $Rej(\mathcal{P}, M)$.

Example 5. Consider the DLP of Ex. 1. By removing rule $a \leftarrow$ from P_1 (due to rule $not\ a \leftarrow$ in P_2) we obtain $\mathcal{P}' = (\{\}, P_2, P_3)$. Note that $\mathcal{P} \overset{DSM}{\equiv} \mathcal{P}'$ and $\mathcal{P} \overset{DJU}{\equiv} \mathcal{P}'$, but $\mathcal{P} \overset{DUM}{\not\equiv} \mathcal{P}'$ and $\mathcal{P} \overset{DAS}{\not\equiv} \mathcal{P}'$. To confirm the later negative case, just observe that $DAS(\mathcal{P}) = \{\{\}, \{a\}\}$ and $DAS(\mathcal{P}') = \{\{\}\}$, i.e. $DAS(\mathcal{P}) \neq DAS(\mathcal{P}')$. Likewise for *DUM.*

5 Discussion and Conclusions

From the definitions and results above, it becomes apparent that the differences between the semantics are very subtle, and all related to the extent that they are immune to tautologies[3]. Of the five semantics presented, the *Refined Dynamic Stable Model* semantics of [1] is the only one that is immune to tautologies. It turns out that such semantics goes a step further and is also immune to more elaborate tautological updates involving dependencies amongst more rules (c.f. [1]).

Related to Logic Program Updates, we can find other semantics in the literature [17, 15], although not following the *causal rejection of rules* approach. They follow a mixture of interpretation updates and rule based updates as they both determine the models of the theory before performing the update, yielding results that differ significantly from the ones described in this paper. In [3] the authors propose Disjunctive Logic Programs with Inheritance, which can be used to encode updates. In [4], the non-disjunctive fragment is proved equivalent to the Update Answer Sets semantics.

References

1. J. J. Alferes, F. Banti, A. Brogi, and J. A. Leite. Semantics for dynamic logic programming: a principle-based approach. In *Procs. of LPNMR-7,* volume 2923 *of LNAI.* Springer, 2004.
2. J. J. Alferes, J. A. Leite, L. M. Pereira, H. Przymusinska, and T. Przymusinski. Dynamic updates of non-monotonic knowledge bases. *Journal of Logic Programming,* 45(1-3), 2000.
3. F. Buccafurri, W. Faber, and N. Leone. Disjunctive logic programs with inheritance. In *Procs. of ICLP'99.* MIT Press, 1999.
4. T. Eiter, M. Fink, G. Sabbatini, and H. Tompits. On properties of update sequences based on causal rejection. *Theory and Practice of Logic Programming,* 2(6), 2002.
5. T. Eiter, M. Fink, G. Sabbatini, and H. Tompits. Using methods of declarative logic programming for intelligent information agents. *Theory and Practice of Logic Programming,* 2(6), 2002.
6. M. Gelfond and V. Lifschitz. Logic programs with classical negation. In *Procs. of ICLP'90.* MIT Press, 1990.
7. H. Katsuno and A. Mendelzon. On the difference between updating a knowledge base and revising it. In .Procs. *of KR'91.* Morgan Kaufmann, 1991.
8. J. A. Leite. *Evolving Knowledge Bases,* volume 81 *of Frontiers in Artificial Intelligence and Applications.* IOS Press, 2003.
9. J. A. Leite and L. M. Pereira. Generalizing updates: From models to programs. In *Procs. of LPKR'97,* volume 1471 of *LNAI.* Springer, 1997.
10. J. A. Leite and L. M. Pereira. Iterated logic program updates. In *Procs. of JICSLP'98.* MIT Press, 1998.
11. N. Leone, G. Pfeifer, W. Faber, F. Calimeri, T. Dell'Armi, T. Eiter, G. Gottlob, G. Ianni, G. Ielpa, S. Perri C. Koch, and A. Polleres. The dlv system. In *Procs. of JELIA'02,* volume 2424 of *LNAI.* Springer-Verlag, 2002.
12. V. Lifschitz and T. Woo. Answer sets in general non-monotonic reasoning (preliminary report). In *Procs. of KR'92.* Morgan-Kaufmann, 1992.

[3] In this paper, for lack of space, we have not explored the classes of programs for which these semantics coincide. In [4,8] the reader can find some results on this subject.

13. V. Marek and M. Truszcczyński. Revision programming. *Theoretical Computer Science,* 190(2), 1998.
14. I. Niemelä and P. Simons. Smodels: An implementation of the stable model and well-founded semantics for normal LP. In *Procs. of LPNMR'97,* volume 1265 of *LNAI.* Springer, 1997.
15. C. Sakama and K. Inoue. Updating extended logic programs through abduction. In *Procs. of LPNMR'99,* volume 1730 of *LNAI.* Springer, 1999.
16. M. Winslett. Reasoning about action using a possible models approach. In *Procs. of AAAI'88,* 1988.
17. Y. Zhang and N. Y. Foo. Updating logic programs. In *Procs. of ECAI'98.* John Wiley & Sons, 1998.

Towards CNC Programming Using Haskell*

G. Arroyo[1], C. Ochoa[2], J. Silva[2], and G. Vidal[2]

[1] CIIDET, Av. Universidad 282 Pte. Centro,
Santiago de Querétaro, Qro, Mexico
garroyo@ciidet.edu.mx
[2] DSIC, Tech. University of Valencia, Camino de Vera s/n,
E-46022 Valencia, Spain
{cochoa, jsilva, gvidal}@dsic.upv.es

Abstract. Recent advances in *Computerized Numeric Control* (CNC) have allowed the manufacturing of products with high quality standards. Since CNC programs consist of a series of assembler-like instructions, several high-level languages (e.g., AutoLISP, APL, OMAC) have been proposed to raise the programming abstraction level. Unfortunately, the lack of a clean semantics prevents the development of formal tools for the analysis and manipulation of programs. In this work, we propose the use of Haskell for CNC programming. The declarative nature of Haskell provides an excellent basis to develop program analysis and manipulation tools and, most importantly, to formally prove their correctness.

1 Introduction

Computerized Numeric Control (CNC for short) machines have become the basis of many industrial processes. CNC machines include robots, production lines, and all those machines that are controlled by digital devices. Typically, CNC machines have a *machine control unit* (MCU) which inputs a CNC program and controls the behavior and movements of all the parts of the machine. Currently—as stated by the standard ISO 6983 [3]—CNC programs interpreted by MCUs are formed by an assembler-like code which is divided into single instructions called *G-codes* (see Fig. 1 below).

One of the main problems of CNC programming is their lack of *portability.* In general, each manufacturer introduces some extension to the standard G-codes in order to support the wide variety of functions and tools that CNC machines provide. Thus, when trying to reuse a CNC program, programmers have to tune it first for the MCU of their specific CNC machines. For example, even though both CNC machines `HASS VF-0` and `DM2016` are milling machines, the G-codes

* This work has been partially supported by CICYT TIC 2001-2705, by Generalitat Valenciana GRUPOS03/025, by EU-India Economic Cross Cultural Prog. \ALA/95/23/2003/077-054, by SEIT/ANUIES under PROMEP-COSNET, and by MCYT under grant HU2003-003.

they accept are different because they belong to different manufacturers (e.g., the former is newer and is able to carry out a wider spectrum of tasks).

CNC programming is not an easy task since G-codes represent a low-level language without control statements, procedures, and many other advantages of modern high-level languages. In order to provide portability to CNC programs and to raise the abstraction level of the language, there have been several proposals of intermediate languages, such as APL [10] and OMAC [8], from which G-codes can be automatically generated with compilers and post-processors. Unfortunately, the lack of a clean semantics in these languages prevents the development of formal tools for the analysis and manipulation of programs. Current CNC programming languages, such as Auto-Code [7] or AutoLISP [2, 13], allow us to completely specify CNC programs but do not permit to analyze program properties like, e.g., termination, or to formally use heuristics when defining the behavior of CNC machines (as it happens with autonomous robots).

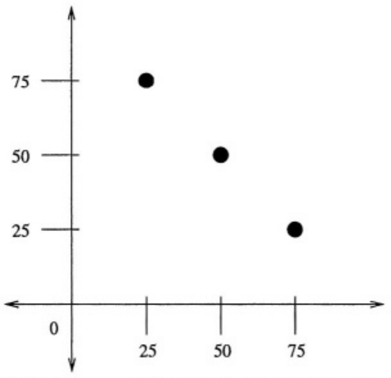
In this work, we propose the use of the *pure* functional language Haskell [12] to design CNC programs. Our choice relies on the fact that Haskell is a modern high-level language which provides a very convenient framework to produce and—formally—analyze and verify programs. Furthermore, it has many useful features such as, e.g., lazy evaluation (which allows us to cope with infinite data structures), higher-order constructs (i.e. the use of functions as first-class citizens, which allows us to easily define complex combinators), type classes for arranging together types of the same kind, etc. This paper presents our proposal for CNC programming using Haskell and shows the main advantages of Haskell over current languages which are used for the same purpose.

This paper is organized as follows. In the next section, we provide a brief review of CNC programming. Section 3 introduces the functional language Haskell. In Sect. 4, we illustrate the use of Haskell to represent CNC programs and, then, in Sect. 5, we enumerate some advantages of choosing Haskell over other existing languages. Some implementation details are discussed in Sect. 6 and, finally, conclusions and some directions for future work are presented in Sect. 7.

2 CNC: A Brief Review

Computer numerical control is the process of having a computer controlling the operation of a machine [19]. CNC machines typically replace (or work in conjunction with) some existing manufacturing processes. Almost all operations performed with conventional machine tools are programmable with CNC machines. For instance, with CNC machines we can perform motion control in linear—along a straight line—or rotary—along a circular path—axes.

A CNC program is composed by series of blocks containing one or more instructions, written in assembly-like format [14]. These blocks are executed in sequential order, step by step. Each instruction has a special meaning, as they get translated into a specific order for the machine. They usually begin with a letter indicating the type of activity the machine is intended to do, like F for feed rate, S for spindle speed, and X, Y and Z for axes motion. For any given CNC



```
N0010  (3-hole drilling)
N0020  G90 G21 G17 G54
N0030  T01 M06
N0040  S1200 M03
N0050  F300
N0060  G00 X25 Y75 Z5
N0070  G01 Z-25
N0080  Z5
N0090  X50 Y50
N0100  Z-25
N0110  Z5
N0120  X75 Y25
N0130  Z-25
N0140  G28 Z0
```

Fig. 1. Simple CNC Program

machine type, there are about 40-50 instructions that can be used on a regular basis. G words, commonly called G codes, are major address codes for preparatory functions, which involves tool movement and material removal. These include rapid moves, lineal and circular feed moves, and canned cycles. M words, commonly called M codes, are major address codes for miscellaneous functions that perform various instructions do not involving actual tool dimensional movement. These include spindle on and off, tool changes, coolant on and off, and other similar related functions. Most G and M-codes have been standardized, but some of them still have a different meaning for particular controllers.

As mentioned earlier, a CNC program is composed by series of blocks, where each block can contain several instructions. For example, `N0030 G01 X3.o Y1.7` is a block with one instruction, indicating the machine to do a movement (linear interpolation) in the X and Y axes. Figure 1 shows an example of a simple CNC program for drilling three holes in a straight line.

3 An Overview of Haskell

Haskell is a general-purpose, purely functional programming language that provides higher-order functions, non-strict semantics, static polymorphic typing, user-defined algebraic datatypes, pattern matching, list comprehensions, a monadic input/output system, and a rich set of primitive datatypes [12]. In Haskell, functions are defined by a sequence of rules of the form

$$f\ t_1\ \ldots\ t_n \quad = \quad e$$

where $t_1, \ldots, t_n$ are *constructor* terms and the right-hand side e is an *expression.*

The left-hand side must not contain multiple occurrences of the same variable. Constructor terms may contain variables and constructor symbols, i.e., symbols which are not defined by the program rules. Functions can also be defined by *conditional equations* which have the form

$$f\ t_1\ \ldots\ t_n \mid c\ =\ e$$

where the condition (or *guard*) c must be a Boolean function. Function definitions have a (perhaps implicit) declaration of its type, of the form

$$f\ ::\ a_1 \rightarrow\ a_2 \rightarrow \ldots \rightarrow a_n \rightarrow b$$

which means that f takes n elements of types $a_1, \ldots, a_n$ and returns an element of type b. For example, the following function returns the length of a given list:

```
length :: [a] -> Int
length []     = 0
length (_:xs) = 1 + length xs
```

Note that in this example, "a" is a *type variable* which stands for any type.

Local declarations can be defined by using the let or where constructs. For example, the following function returns True if the first argument is smaller than the length of the list being passed as a second argument, or False otherwise:

```
indexChecker :: Int -> [a] -> Bool
indexChecker n xs = n <= l where l = length xs
```

A Haskell function is *higher-order* if it takes a function as an argument, returns a function as a result, or both. For instance, map is a higher-order function that applies a given function to each element in a list:

```
map :: (a -> b) -> [a] -> [b]
map f []     = []
map f (x:xs) = f x : map f xs
```

Haskell provides a static type semantics, and even though it has several primitive datatypes (such as integers and floating-point numbers), it also provides a way of defining our own (possibly recursive) types using data declarations:

```
data Bool   = False  | True
data Tree a = Leaf a | Branch (Tree a) (Tree a)
```

For convenience, Haskell also provides a way to define *type synonyms;* i.e., names for commonly used types. Type synonyms are created using a type declaration. Here are some examples:

```
type String = [Char]            type Name    = String
type Person = (Name,Address)  data Address = None | Addr String
```

Type classes (or just classes) in Haskell provide a structured way to introduce *overloaded* functions that must be supported by any type that is an *instance* of that class. For example, the Equality class in Haskell is defined as follows:

```
class Eq a where
   (==) :: a -> a -> Bool
```

G01: Moves the turret chuck along the XYZ axes. It can be followed by XYZ codes.
G90: Indicates that absolute positioning is being used.
G91: Indicates that incremental positioning is being used.
    X(-)nn: used to move the turret chuck along the X axis
    Y(-)nn: used to move the turret chuck along the Y axis
    Z(-)nn: used to move the turret chuck along the Z axis

where **nn** indicates:

- the new absolute position in the corresponding axis, where (0,0,0) is a given reference point over the table (*absolute* positioning).
- the number of units in the current axis that the tool is being shifted (*incremental* positioning).

Fig. 2. Instructions set for simple CNC drilling machine

A type is made an instance of a class by defining the signature functions for the type, e.g., in order to make `Address` an instance of the `Equality` class:

```
instance Eq Address where
   None      == None       = True
   Addr st1 == Addr st2  = st1 == st2
   _         == _          = False
```

where _ is a wildcard, used to introduce a default case.

We refer the interested reader to the report on the Haskell language [12] for a detailed description of all the features of the pure functional language Haskell.

4 Using Haskell for CNC Programming

In this section, we illustrate the use of Haskell for CNC programming. By lack of space, we consider a *simple* CNC drilling machine which can just move the turret chuck in the X and Y axes (in order to position the drill bit), and in the Z axis (in order to make the hole). The machine also handles absolute and incremental positioning of the turret chuck.[1]

A CNC program for this machine consists of a header and a body. The header is optional and is usually a short comment, whilst the body is a list of blocks, where each block is identified by a number `(Nnnnn)` and can contain either one or more instructions or a comment, where comments are always parenthesized.

An instruction can contain one of the CNC codes shown in Fig. 2. In the following, we consider millimeters as measurement units.

For instance, a CNC program for drilling two holes at positions (10,10) and (15,15) is as follows:

[1] A full example with the implementation of complete data structures needed to represent CNC programs can be found at `http://www.dsic.upv.es/~jsilva/cnc`.

```
data CNCprogram  = Header Body
data Header      = Maybe Comment
type Comment     = String
data Maybe a     = Nothing | Just a
type Body        = [Block]
data Block       = Com Comment | Code Command
type Command     = [Instruction]
data Instruction = X Int | Y Int | Z Int | G String
```

Fig. 3. Haskell data structures for representing a CNC program

```
N0010 (two-hole drilling)          N0060 Z05
N0020 G90                          N0070 X15 Y15
N0030 G01 Z05                      N0080 Z-25
N0040 X10 Y10                      N0090 Z05
N0050 Z-25
```

In this example, the block `N0010` denotes a comment, `N0020` instructs the CNC machine to use absolute positioning, `N0030` moves the turret chuck 5mm over the table in the Z axis, `N0040` positions the turret chuck in the (10,10) coordinate (note that `X10 Y10` is a shortcut for `G01 X10 Y10`), `N0050` moves the turret chuck 25mm under the table in the Z axis, thus making a hole, `N0060` moves the turret chuck 5mm over the table in the Z axis, in order to be able to move it again in the XY axes, `N0070` positions the turret chuck in the (15,15) coordinate, and finally `N0080` and `N0090` create the second hole.

It should be clear from this example that, when a big number of holes should be done, the amount of lines of code we have to write also increases considerably. The Haskell data structure intended to hold a CNC program is shown in Fig. 3.

For simplicity, our data structure does not contain information about block numbering. Nevertheless, given a list of blocks, it is straightforward to build a function that adds such numbering to each block. In our context, a CNC program is composed of a header and a body. A *header* is an optional comment (a `String` or the constructor `Nothing` when missing). A *body* is a set of blocks, each block being a comment or a set of instructions.

Figure 4 shows a function for drilling *n* holes in a straight line implemented in Haskell. Note that `nHolesLine` returns a list of blocks, instead of a complete CNC program as defined in Fig. 3 (the header is missing). Far from being a shortcoming, this is due to the fact that `nHolesLine` is integrated in an environment with many other different functions; therefore, there is a master function which builds the whole program by prompting the user for a header comment, if any, and integrating the code obtained from the different functions.

The function `nHolesLine` receives as parameters the number of holes, `n`, the initial XY coordinate and the corresponding increments in each axis. Then, this function generates a list of blocks by creating a list of `n` elements containing the *absolute* XY coordinates of the holes; it uses the higher-order function `map` to apply the function `makeHole` to each XY coordinate.

```
-- creates a [Block] containing the CNC instructions for
-- making n holes in a straight line
nHolesLine :: Int -> Int -> Int -> Int -> Int ->  [Block]
nHolesLine n x y incX incY = [posit,init] ++ concat (map makeHole nLine)
  where line    = createLine x y incX incY
        nLine   = finiteLine n line
        posit   = Code [G "90"]
        init    = Code [G "00",Z 5]

-- creates an infinite list of absolute coordinates
createLine :: Int -> Int -> Int -> Int -> [(Int,Int)]
createLine x y incX incY = (x,y):createLine (x+incX) (y+incY) incX incY

-- takes a list and returns a sublist containing the first n elements
finiteLine :: Int -> [a] -> [a]
finiteLine _ []     = []
finiteLine 0 _      = []
finiteLine n (x:xs) = x : finiteLine (n-1) xs

-- takes an XY coordinate and return a block containing
-- all instructions needed to make a hole at such position
makeHole :: (Int,Int) -> [Block]
makeHole (x,y) = [posXY,down,up]
   where up    = Code [Z 5]
         down  = Code [Z (-25)]
         posXY = Code [X x,Y y]
```

Fig. 4. Example program `nHolesLine`

Note that, even though the list of `n` elements is created by first calling `createLine`—which generates an *infinite* list—only `n` elements of such a list are actually built. This is due to the lazy evaluation of Haskell (see Sect. 5). For instance, in order to make 50 holes, starting at position (10,10) and increasing 5mm in each axis per hole, we simply call function `nHolesLine` as follows:

```
nHolesLine 50 10 10 5 5
```

The same example using G codes requires about 150 lines of code. However, with Haskell functions it is very simple to change any of the parameters to achieve any straight line of holes to be drilled. In the same way, a lot of helpful functions can be created, in order to make different shapes like ellipses, grids, etc, that are able to work with other CNC machines like lathes and milling machines.

5 Some Advantages of Haskell

In the following, we summarize the most significant advantages of Haskell for CNC programming:

Data Structures and Recursion. Haskell allows the definition of complex data structures (such as 3D geometric pieces) or iterative ones (such as hole meshes) that can be later manipulated and reused. A common way of defining and manipulating such data structures is to use recursion. In Fig. 4, two lists, `line` and `nLine,` are used to describe a set of specific positions of a piece. We recursively apply a defined function to them by using a simple command.

Polymorphism. Functions in Haskell can be polymorphic, i.e., they can be applied to different types (compare function `length`, which can be applied to lists of any kind). In our context, this means that some functions can be reused in many parts of the CNC program with different input data.

Higher-Order Functions. Higher-order facilities [4] are one of the main advantages of Haskell over the rest of languages currently used for CNC programming, incorporating a big amount of predefined higher-order functions allowing us to optimize and minimize the size of the code with a high expressiveness.

Laziness. Haskell follows a *lazy* evaluation model [1, 18], which means that functions are evaluated *on demand.* This is particularly useful when dealing with infinite data structures. For instance, consider a robot hand which performs a specific movement each time a piece is under it. Thanks to laziness, we can define the behavior of the robot hand by this infinite movement since it will only be evaluated as much as needed. To the best of our knowledge, all languages used in CNC programming lack of lazy evaluation (i.e., they are strict languages with *call by value* evaluation).

Type Checking System. Haskell includes a standard type inference algorithm during compilation. Type checking can be very useful to detect errors in a CNC program, e.g., to detect that a drilling tool has not been separated from the piece surface after its use. Since CNC programs are usually employed in mass-production processes, program errors are very expensive. Therefore, having built-in type error checkers provided by a high-level compiler represents a significant advantage over usual CNC programs written by hand.

Type Classes. Type classes in Haskell provide a structured way to introduce *overloaded*[2] functions that must be supported by any type that is an *instance* of that class [20]. This is a powerful concept that allows us, e.g, to arrange together data structures (representing CNC machines) having a similar functionality and to define standard functions for such types. When introducing a new data structure representing a CNC machine having a functionality similar to an existing one, we can just derive it from such a class, applying the existing functions to this data structure without re-writing any code.

Verification and Heuristics. Haskell is a formal language with many facilities to prove the correctness of programs [15]. This represents the main advantage of

[2] While polymorphic functions have a *single* definition over many types, overloaded functions have *different* definitions for different types.

our proposal compared with current languages being used for the same purpose. Thus, formal verification of CNC programs can be performed to demonstrate its termination, correctness, etc. Moreover, it makes possible the application of heuristics to define the behavior of CNC machines (as it happens with autonomous robots). This is subject of ongoing work and justifies our choice of Haskell for CNC programming.

Furthermore, Haskell is amenable to formal verification by using theorem provers such as Isabelle [11] or HOL [9, 17], verification logics such as P-Logic [5,6], etc.

6 Implementation Remarks

The implementation of a Haskell library to design and manipulate CNC programs has been undertaken. This library currently contains:

– an XML DTD properly defined to completely represent any CNC program,
– a specific Haskell data structure equivalent to the DTD, and
– a set of functions to build and test CNC programs.

In order to guarantee the portability of the CNC programs produced in our setting, we have defined an XML DTD which is able to represent any CNC program since it contains all the syntactic constructs specified in the standard ISO 6983 [3], as well as the extensions proposed in [16, 19]. With this DTD, we can properly define any CNC program and, with some Haskell translation functions, automatically convert it to/from an equivalent Haskell data structure.

We have also implemented a library of functions which allows us to build and transform the data structure representing CNC programs in a convenient way. We provide several basic testing and debugging functions. Preliminary experiments are encouraging and point out the usefulness of our approach. More information (the implementation of Haskell library, the XML DTD and some examples) are publicly available at `http://www.dsic.upv.es/~jsilva/cnc`

7 Conclusions and Future Work

This work proposes Haskell as a high-level language for the design and implementation of CNC programs. We have clarified its advantages over existing languages for CNC programming. Besides typical high-level features—such as control sequence, recursion, rich data structures, polymorphism, etc.—Haskell provides several advanced features like higher-order combinators, lazy evaluation, type classes, etc. Furthermore, Haskell offers a clean semantics which allows the development of formal tools.

We have defined a Haskell data structure which is able to represent any CNC program, allowing us to properly manipulate it by using Haskell features. Furthermore, we implemented an XML DTD which is fully equivalent to the

Haskell data structure. This DTD ensures the portability of our programs among applications and platforms.

Preliminary experiments are encouraging and point out the usefulness of our approach. However, there is plenty of work to be done, like augmenting our library with other useful functions for making geometric figures, defining functions for other CNC machines (lathes, milling machines, etc), defining libraries for assisting the user in the post-processing of CNC programs, defining a graphical environment for simplifying the task of designing CNC programs, etc.

References

1. R. Bird. *Introduction to Functional Programming Using Haskell, 2nd Ed.* Prentice Hall Press, 1998.
2. H. Carr and R. Holt. The AutoLISP Platform for Computer Aided Design. In *40th Anniversary of Lisp Conference: Lisp in the Mainstream,* Berkeley, California, November 1998.
3. International Standardization for Organizations. Technical committee: ISO/TC 184/SC 1. Numerical control of machines – Program format and definition of address words, September 1982.
4. J. Hughes. Why Functional Programming Matters. *Computer Journal,* 32(2):98–107, 1989.
5. R. Kieburtz. P-logic: Property Verification for Haskell Programs, 2002. The Programatica Project, http://www.cse.ogi.edu/PacSoft/projects/programatica/.
6. R. Kieburtz. Automated Soundness Checking of a Programming Logic for haskell, 2003. The Programatica Project, available at http://www.cse.ogi.edu/PacSoft/projects/programatica/.
7. B. Kramer. *The AutoCADET's Guide to Visual LISP.* CMP Books, 2001.
8. J. Michaloski, S. Birla, C.J. Yen, R. Igou, and G. Weinert. An Open System Framework for Component-Based CNC Machines. *ACM Computing Surveys,* 32(23), 2000.
9. T.F. Melham M.J.C. Gordon. *TIntroduction to HOL.* Cambridge University Press, 1993.
10. T.P. Otto. An apl compiler. In International Conference on APL, editor, *Proceedings of the international conference on APL-Berlin-2000 conference,* pages 186–193. ACM Press - New York, NY, USA, 2000.
11. L.C. Paulson. The Foundation of a Generic Theorem Prover. *Journal of Automated Reasoning,* 5(3):363–397, 1989.
12. S. Peyton Jones, editor. *Haskell 98 Language and Libraries: The Revised Report.* Cambridge University Press, 2003.
13. R. Rawls and M. Hagen. *AutoLISP Programming: Principles and Techniques.* Goodheart-Willcox Co, 1998.
14. W. Seames. *CNC: Concepts and Programming.* Delmar Learning, 1994.
15. S. Thompson. Formulating Haskell. Technical Report 29-92*, University of Kent, Computing Laboratory, University of Kent, Canterbury, UK, November 1992.
16. J. Richard W. Maeder, V. Nguyen and J. Stark. Standardisation of the Manufacturing Process: the IMS STEP-NC Project. In *Proc. of the IPLnet Workshop,* 2002.
17. Tanja E.J. Vos. Inductive Datatypes with Negative Occurrences in HOL. Workshop on Thirty Five years of Automath, Edinburgh, UK, 2002.

18. P. Wadler. The Essence of Functional Programming. In *Proceedings of the Symposium on Principles of Programming Languages (POPL '92), Albuquerque.* ACM Press, 1992.
19. M. Weck, J. Wolf, and D. Kiritsis. Step-nc The STEP Compliant NC Programming Interface: Evaluation and Improvement of the Modern Interface. In *Proc. of the ISM Project Forum 2001,* 2001.
20. M. Wenzel. Type Classes and Overloading in Higher-Order Logic. In E. Gunter and A. Felty, editors, *Proceedings of the 10th International Conference on Theorem Proving in Higher Order Logics (TPHOLs'97),* pages 307–322, Murray Hill, New Jersey, 1997.

Well Founded Semantics for Logic Program Updates

F. Banti[1], J. J. Alferes[1], and A. Brogi[2]

[1] CENTRIA, Universidade Nova de Lisboa, Portugal
[2] Dipartimento di Informatica, Università di Pisa, Italy

Abstract. Over the last years various semantics have been proposed for dealing with updates of logic programs by (other) logic programs. Most of these semantics extend the stable models semantics of normal, extended (with explicit negation) or generalized (with default negation in rule heads) logic programs. In this paper we propose a well founded semantics for logic programs updates. We motivate our proposal with both practical and theoretical argumentations. Various theoretical results presented here show how our proposal is related to the stable model approach and how it extends the well founded semantics of normal and generalized logic programs.

1 Introduction

When dealing with knowledge bases modelling knowledge that may change over time, an important issue is that of how to automatically incorporate new (updated) knowledge without falling into an inconsistency each time this new knowledge is in conflict with the previous one. When knowledge is represented by logic programs (LPs), this issue boils down to that of how to deal with LPs updates. In this context, updates are represented by sequences of sets of logic programming rules, also called *dynamic logic programs* (DyLPs), the first set representing our initial knowledge, while later ones represent new incoming information. In the last years, several semantics had been proposed for logic programs updates [1,2,5,9,13–15,17,18]. Most of these semantics are extensions of the stable models semantics of extended (with explicit negation) [12] or generalized (allowing default negation in rule heads) [16] LPs. This is a natural choice given the appropriateness of stable models for knowledge representation, and the simplicity of the definition of stable model semantics for normal LPs, which allows various extensions in a natural way. However, it is our stance that there are application domains for logic programs updates with requirements demanding a different choice of basic semantics, such as the well founded semantics [11]. One of such requirements is that of computational complexity: in applications that require the capability of dealing with an overwhelming mass of information, it is very important to be able to quickly process such information, even at the cost of losing some inference power. In this respect, as it is well known, the computation of stable models is NP-hard, whereas that of the well founded model is polynomial. Another requirement not fulfilled by stable model semantics is that of being able to answer queries about a given part of the knowledge without the need to, in general, consult the whole knowledge base. The well founded semantics complies with the property of relevance [8], making it possible to implement query driven proof procedures that, for any given query, only need to explore a part of the knowledge base. Moreover, in domains with a great amount of highly distributed and heterogeneous knowledge, inconsistencies are bound to appear not only when new knowledge conflicts with old knowledge, but also within the new (or old) knowledge alone. To deal with contradictions that appear simultaneously, the mechanisms of updates are of no use, and some form of paraconsistent semantics [7] is required, i.e. a semantics where these contradictions are at least detected, and isolated. A well founded based semantics for LPs updates seems to be the answer for domains where the above requirements are added with the need to update knowledge. However, as we mentioned above, most of the existing semantics are stable models based. A few attempts to define a well founded semantics for DyLPs can be found [2, 3, 13]. Unfortunately, as discussed in Section 5.1, none of these is, in our opinion, satisfactory, be it because they lack a declarative definition of the semantics, or because they are too skeptical.

In this paper we define the (paraconsistent) well founded semantics of DyLPs. This semantics is a generalization for sequences of programs of the well founded semantics of normal [11] and generalized LPs [6]. Moreover it is sound wrt to the stable models semantics for DyLPs as defined in [1]. As for most of the existing semantics for DyLPs, the approach herein is also based on the causal rejection principle [9, 14], which states, informally: an old rule is rejected if there exists a more recent one which is supported and whose immediate conclusions are in conflict with the ones of the older rule. We extend this principle from a 2-valued to a 3-valued setting, and apply it to the well founded semantics.

The rest of the paper is organized as follows. Section 2 recalls some preliminary notions and establishes notation. Section 3 presents the extension of the causal rejection principle to the 3-valued case. In section 4 the well founded semantics for DyLPs is defined, and in section 5 some of its properties are studied and relations with existing proposals (briefly) established. We end, in section 6, with some concluding remarks.

2 Background: Language, Concepts and Notation

In this section we briefly recall the syntax of DyLPs, a language introduced in [2] for dealing with logic programs updates, and their semantics as defined in [1]. Our choice on this semantics for introducing the background is based on the fact that, among the existing ones, it is the more credulous and that it properly overcomes some problems of the existing ones, as shown in [1].

To represent negative information in logic programs and their updates, DyLP uses generalized logic programs (GLPs) [16], which allow for default negation *not A* not only in the premises of rules but also in their heads. A GLP defined over a propositional language $\mathcal{L}$ is a (possibly infinite) set of ground rules of the form $L_0 \leftarrow L_1, \ldots, L_n$, where each L_i is a literal in $\mathcal{L}$, i.e., either a propositional atom A in $\mathcal{L}$ or the default negation *not A* of a propositional atom A in $\mathcal{L}$. We say that A is the *default complement* of *not A* and viceversa. Given a rule τ as above, by $hd(\tau)$ we mean L_0 and by $B(\tau)$ we mean $\{L_1, \ldots, L_n\}$. In the sequel an *interpretation* is simply a set of literals of $\mathcal{L}$. A literal L is *true* (resp. *false*) in I iff $L \in I$ (resp. $not\, L \in I$) and *undefined* in I iff $\{L, not\, L\} \cap I = \{\}$. A conjunction (or set) of literals C is true (resp. false) in I iff $C \subseteq I$ (resp. $\exists\, L \in C$ such that L is false in I). We say that I is *consistent* iff $\forall\, A \in \mathcal{L}$ *at most* one of A and *not A* belongs to I, otherwise we say I is *paraconsistent.* We say that I is *2-valued* iff for each atom $A \in \mathcal{L}$ *exactly* one of A and *not A* belongs to I.

A *dynamic logic program* over a language $\mathcal{L}$ is a finite sequence $P_1 \oplus \ldots \oplus P_n$ (also denoted $\oplus P_i$, where the P_is are GLPs indexed by $1, \ldots, n$), where all the P_is are defined over $\mathcal{L}$. Intuitively such a sequence may be viewed as the result of, starting with program P_1, updating it with program P_2, ..., and updating it with program P_n. For this reason we call the singles $P_i s$ *updates.* We use $\rho(\mathcal{P})$ to denote the multiset of all rules appearing in the programs $P_1, ..., P_s$.

The *refined stable model semantics* for DyLPs is defined in [1] by assigning to each DyLP a set of stable models The basic idea of the semantics is that, if a later rule τ has a true body, then former rules in conflict with τ should be *rejected* . Moreover, any atom A for which there is no rule with true body in any update, is considered false by default. The semantics is then defined by a fixpoint equation that, given an interpretation I, tests whether I has exactly the consequences obtained after removing from the multiset $\rho(\mathcal{P})$ all the rules rejected given I, and imposing all the default assumptions given I. Formally, let:

$$Default(\oplus P_i, I) = \{not\ A \mid \nexists\ A \leftarrow body \in \rho(\mathcal{P}) \wedge body \subseteq I\}$$
$$Rej^S(\oplus P_i, I) = \{\tau \mid \tau \in P_i \mid \exists\ \eta \in P_j\ i \leq j,\ \tau \bowtie \eta \wedge\ B(\eta) \subseteq I\}$$

where $\tau \bowtie \eta$ means that τ and η are conflicting rules, i.e. the head of τ is the default complement of the head of η.

Definition 1. *Let $\oplus P_i$ be a DyLP overlanguage $\mathcal{L}$ and M a two valued interpretation. M is a refined stable model of $\oplus P_i$ iff M is a fixpoint of $\Gamma^S_{\oplus P_i}$:*

$$\Gamma^S_{\oplus P_i}(M) = least\left(\rho(\mathcal{P}) \setminus Rej^S(\oplus P_i, M) \cup Default(\oplus P_i, M)\right)$$

where least(P) denotes the least Herbrand model of the definite program obtained by considering each negative literal not A in P as a new atom[1].

The definition of dynamic stable models of DyLPs [2] is as the one above, but where the $i \leq j$ in the rejection operator is replaced by $i < j$. I.e., if we denote this other rejection operator by $Rej(\oplus P_i, I)$, and define $\Gamma_{\oplus P_i}(I)$ by replacing in Γ^S Rej^S by *Rej,* then the stable models of $\oplus P_i$ are the interpretations I

[1] Whenever clear from the context, hereafter we omit the $\oplus P_i$ in any of the above defined operators.

such that $I = \Gamma_{\oplus P_i}(I)$. Comparisons among these two definitions, as well as further details, properties and motivation for the definition of this language and semantics are beyond the scope of this paper, and can be found in [1, 2].

3 The Notion of Causal Rejection for 3-Valued Semantics

According to the above mentioned causal rejection principle [9, 14], a rule from an older program in a sequence is kept (by inertia) unless it is rejected by a more recent conflicting rule whose body is true. On the basis of this, the very basic notion of model has to be modified when dealing with updates. In the static case, a model of a program is an interpretation that satisfies all the rules of the program, where a rule is satisfied if its head is true or its body is false. If we want to adapt this idea to the updates setting taking in consideration the casual rejection principle we should only require non rejected rules to be satisfied. Also the concept of supported model [4] has to be revisited when dealing with updates. In the static case, a model M of P is supported iff for every atom $A \in M$, there is a rule in P whose head is A and whose body is satisfied in M. If we extend the concept of supportedness to logic programs with updates, it would be unnatural to allow rejected rules to support a the truth of a literal.

The causal rejection principle is defined for 2-valued semantics; we want now to extend it to a 3-valued setting, in which literals can be *undefined,* besides being *true* or *false.* In the 2-valued setting, a rule is rejected iff there is a rule in a later update whose body is true in the considered interpretation. In this context, this is the same as saying that the body of the rejecting rule is not false. In a 3-valued setting this is no longer the case, and the following question arises: should we reject rules on the basis of rejecting rules whose body is *true,* or on the basis of rules whose body is *not false*? We argue that the correct answer is the latter. In the remainder we give both practical and theoretical reasons for our choice, but we want now to give an intuitive justification. Suppose initially we believe a given literal L is true. Later on we get the information that L is false if some conditions hold, but those conditions are (for now) undefined. As usual in updates, we prefer later information to the previous one. On the basis of such information, can we be *sure* that L remains true? It seems to us we cannot. The more recent source of information says if some conditions hold then L is false, and such conditions *may* hold. We should then reject the previous information and consider, on the basis of the most recent one, that L is undefined.

On the basis of these intuitions, we extend the definition of update model and update supported model to the 3-valued setting.

Definition 2. *Let $\oplus P_i$ be any DyLP, and M a 3-valued interpretation. M is an* update 3-valued model *of $\oplus P_i$ iff for each rule τ in any given P_i, M satisfies τ (i.e. $hd(\tau) \in M$ or $B(\tau) \not\subseteq M$) or there exists a rule η in P_j, $i < j$ such that $\tau \bowtie \eta$ and $B(\eta)$ is not false in M. We say M is a* supported 3-valued update model *of $\oplus P_i$ iff it is an update 3-valued model and*

1. *for each atom $A \in M$, $\exists\ \tau \in P_i$ with head A such that $B(\tau) \subseteq M$ and $\nexists$ $\eta \in P_j$, $i < j$ such that $\tau \bowtie \eta$, and $B(\eta)$ is not false in M.*
2. *for each negative literal not A, if not $A \in M$, then for each rule $A \leftarrow body \in \rho(\mathcal{P})$ such that body is true in M, there exists a rule η, in a later update whose head is not A, and such that $B(\eta)$ is true in M.*

We illustrate, via an example, the intuitive meaning of the defined concepts.

Example 1. Sara, Cristina and Bob, are deciding what they will do on Saturday. Sara decides she is going to a museum, Cristina wants to go shopping and Bob decides to go fishing in case Sara goes to the museum. Later on they update their plans: Cristina decides not to go shopping, Sara decides she will not go to the museum if it snows and Bob decides he will also go fishing if it is a sunny day. Moreover we know from the forecast that Saturday can be either a sunny day or a raining day. We represent the situation with the DyLP $P_1 \oplus P2$, where:

$$\begin{array}{lll}
P_1 : museum(s). & P_2 : fish(b) \leftarrow sunny. & sunny \leftarrow not\, rain. \\
\quad shopping(c). & \quad not\, shopping(c). & rain \leftarrow not\, sunny. \\
\quad fish(b) \leftarrow museum(s). & \quad not\, museum(s) \leftarrow snow. &
\end{array}$$

The intended meaning of $P_1 \oplus P_2$ is that it does not snow on Saturday, but we do not know if it does rain or not, we know Sara goes to the museum on Saturday, Bob goes fishing and, finally, Cristina does not go shopping. In fact, every 3-valued update model of $P_1 \oplus P_2$ contains $\{museum(s), not\, shopping(c), fish(b)\}$. Suppose now Bob decides that, in the end, he does not want to go fishing if it rains, i.e our knowledge is updated with: P_3 : $not\, fish(b) \leftarrow rain$. Intuitively, after P_3, we do not know whether Bob will go fishing since we do not know whether Saturday is a rainy day. According to definition 2, there is a supported 3-valued update model of $P_1 \oplus P_2 \oplus P_3$ in which $shopping(c)$ is false, $museum(s)$ is true and $fish(b)$ is undefined.

It can be checked that, according to all existing stable models based semantics for updates of [1,2,5,9,14,15], $P_1 \oplus P_2 \oplus P_3$ has two stable models: one where *rain* is true and $fish(b)$ is false, and another where *rain* is false and $fish(b)$ is true. A notable property of the well founded model in the static case is that of being a subset of all stable models. If one wants to preserve this property in DyLPs, in the well founded model of this example one should neither conclude $fish(b)$ nor $not fish(b)$. If a rule would only be rejected in case there is a conflicting later one with *true* body (rather than *not false* as we advocate), since the body of $not\, fish(b) \leftarrow rain$ is not true, we would not be able to reject the initial rule $fish(b) \leftarrow museum(s)$, and hence would conclude $fish(b)$. Hence, to preserve this relation to stable models based semantics, the well founded model semantics for DyLPs must rely on this notion of 3-valued rejection described above.

4 The Well Founded Semantics for DyLPs

On the basis of the notion of causal rejection just presented, we define the Well Founded Semantics for DyLPs. Formally, our definition is made in a way similar to the the definition of the well founded semantics for normal LPs in [10], where the well founded model is characterized by the least alternating fixpoint of the Gelfond-Lifschitz operator Γ (i.e. by the least fixpoint of Γ^2). Unfortunately, if we apply literally this idea, i.e. define the well founded model as the least alternating fixpoint of the operator used for the dynamic stable (or refined stable) models of DyLPs, the resulting semantics turns out to be too skeptical:

Example 2. It is either day or night (but not both). Moreover, if the stars are visible it is possible to make astronomical observations. This knowledge is updated with the information that: if it is night the stars are visible; the observatory is closed if it is not possible to make observations; and the stars are not visible:

$P_1 : observe \leftarrow see\_stars.$	$day \leftarrow not\, night.$	$night \leftarrow not\, day.$
$P_2 : see\_stars \leftarrow night.$	$not\, see\_stars.$	$closed(obs) \leftarrow not\, observe.$

The intended meaning of $P_1 \oplus P_2$ is that currently the stars are not visible, it is not possible to make astronomical observations and, hence, the observatory is closed. However, it is easy to check, the least alternating fixpoint of $\Gamma_{P_1 \oplus P_2}$ is *{not see_stars},* in which one is not able to conclude that the observatory is closed. This is, in our opinion, not satisfactory: since we conclude that we cannot see the stars, we should also conclude that we cannot make astronomical observations and that the observatory is closed. Notably, the least alternating fixpoint of Γ^S yield even more skeptical results. In fact, this is a general result which is an immediate consequence of Lemma 1 below.

In order to overcome this problem, we define it as the least fixpoint of the composition of two different (antimonotonous) operators. Such operators have to deal with the causal rejection principle described above, in which a rule is to be rejected in case there is a later conflicting one whose body is *not false.* In the well founded semantics of normal logic programs, if there exists a rule $A \leftarrow body$ (where *A is* an atom), such that *body* is not false in the well founded model, then *A* is not false as well. Consider now the same situation in an update setting with a rule $L \leftarrow body_1$, where $body_1$ is not false. In this situation we should conclude that *L* is not false *unless* there exists a rule $not\, L \leftarrow body_2$, where $body_2$ is true in the same or in a later program in the sequence. In fact, note that the rule for *not L* is not rejected by the one for *L.* Since the body of the former is true, according to the causal rejection principle *not L* should be true (i.e. *L* should be false) unless the rule is rejected by some later rule. In any case, $L \leftarrow body_1$ is no longer playing any role in determining the truth value of *L.* For this reason we allow rules to reject other rules in previous *or in the same* update while determining the set of non-false literals of the well founded model and, accordingly, we use $\Gamma^S_{\oplus P_i}$, as the first operator of our composition.

For determining the set of true literals according to the causal rejection principle, only the rules that are not rejected by conflicting rules in *later* updates should be put in place. For this reason we use $\Gamma_{\oplus P_i}$ as the second operator, the well founded model being thus defined as the least fix point of the $\Gamma\Gamma^S$.

Definition 3. *The well founded model $WFDy(\oplus P_i)$ of a DyLP $\oplus P_i$ is the (set inclusion) least fixpoint of $\Gamma_{\oplus P_i}\Gamma^S_{\oplus P_i}$.*

Since both Γ and Γ^S are antimonotonous (cf. [1,14]), $\Gamma\Gamma^S$ is monotonous, and so it always has a least fixpoint. In other words, *WFDy* is uniquely defined for every DyLP. Moreover $WFDy(\oplus P_i)$ can be obtained by (transfinitely) iterating $\Gamma\Gamma^S$, starting from the empty interpretation.

For the dynamic logic program $P_1 \oplus P_2$ of example 2, the well founded model is *{not see_stars, not observe, closed(obs)}*. So, in this example, *WFDy* yield the desired less skeptical conclusions. In fact, in general *WFDy* is less skeptical than any semantics resulting from any other combination of Γ^S and Γ.

Lemma 1. *Let $\oplus P_i$ be a DyLP, and let X, Y be two interpretations such that $X \subseteq Y$. Then $\Gamma^S(Y) \subseteq \Gamma(X)$.*

From this Lemma it follows that the least fixpoint of any other combination is a pre-fixpoint of $\Gamma\Gamma^S$ and, as such, a subset of the least fixpoint of $\Gamma\Gamma^S$.

Example 3. Let P_1, P_2 and P_3 be the programs of example 1. As desired, the well founded model of $P_1 \oplus P_2 \oplus P_3$ is, $\{not\ snow,\ museum(s), not\ shopping(c)\}$.

As shown in example 1, $WFDy(P_1 \oplus P_2 \oplus P_3)$ is a supported 3-valued update model. This result holds in general, whenever the well founded model does not contain any pair of complementary literals.

Theorem 1. *Let $\oplus P_i$ be a DyLP and W its well founded model. Then, if W contains no pair of complementary literals, W is a supported 3-valued update model of $\oplus P_i$.*

The proviso of *W* not containing any pair of complementary literals is due to the fact that, since notion of interpretation we use allows contradictory sets of literals, the well founded model of a DyLP can be contradictory. We say that a DyLP $\oplus P_i$ is consistent (or non contradictory) iff $WFDy(\oplus P_i)$ is consistent i.e. it does not contain any pair of complementary literals.

5 Properties

Our motivation for defining a new semantics for logic programs updates, as described in the Introduction, is based on a number of requirements. Hence, we briefly examine in what term those requirements are indeed met by *WFDy,* and briefly comparing with existing approaches.

One of the important requirements is that of having a semantics computable in polynomial time. It is not difficult to check that the computation of both $\Gamma_{\oplus P_i}(I)$ and $\Gamma^S_{\oplus P_i}(I)$ is polynomial in the size of *DyLP,* and so:

Proposition 1. *The well founded model of any finite ground dynamic logic program $\oplus P_i$ is computable in polynomial time on the number of rules in $\oplus P_i$.*

Another required property is that of relevance [8], so as to guarantee the possibility of defining query driven proof procedures. Informally, in normal (single) programs, a semantics complies with relevance if the truth value of any atom A in a program only depends on the rules relevant for this literal (i.e. those rules with head A, or with a head A' such that A' belongs to the body of (another) relevant rule). In order to establish results regarding relevance of *WFDy* we have first to define what is the relevant part of a DyLP (rather than a single program) wrt a literal (rather than atom).

Definition 4. *Let $\oplus P_i$ be any DyLP in the language $\mathcal{L}$ and L, B, C literals in $\mathcal{L}$. We say L directly depends on B iff B occurs in the body of some rule in $\oplus P_i$ with head L or not L. We say L depends on B iff L directly depends on B or there is some C such that L directly depends on C and C depends on B. We call $Rel_L(\oplus P_i)$ the dynamic logic programs $P_1^{(L)} \oplus \ldots \oplus P_n^{(L)}$ such that $P_i^{(L)}$ is the set of all rules of P_i with head L or not L or some B on which L depends on.*

This definition simply applies the above intuition of relevance considering sequences of programs, and by stating that rules for *not A* are relevant for A (and vice-versa). And *WFDy* complies with relevance exactly in these terms:

Theorem 2. *Let $\oplus P_i$ be a DyLP in the language $\mathcal{L}$ and A any atom of $\mathcal{L}$. Then $WFDy(\oplus P_i) \cap \{A,\ not\, A\} = WFDy(Rel_A \oplus P_i) \cap \{A,\ not\, A\}$.*

As noted above, *WFDy* can be contradictory. In these cases, inconsistent conclusions for a given atom may follow, but without necessarily having a contradiction in all atoms. However, as desired in updates, these contradictions in atoms may only appear in case there are two conflicting simultaneous rules (i.e. in a same program of the sequence) which are both supported, and none of them is rejected by some later update:

Theorem 3. *The well founded model W of a sequence $\oplus P_i$ is noncontradictory iff for all $\tau, \eta \in P_i$ such that: $\tau \bowtie \eta$, $W \models B(\tau)$, $W \models B(\eta)$) there exists $\gamma \in P_j$, $i < j$ such that $\gamma \bowtie \tau$ or $\gamma \bowtie \eta$ and $\Gamma^S(W) \models B(\gamma)$.*

Finally, it was our goal to find a proper generalization of the well founded semantics single programs into logic programs updates. It is thus important, to guarantee that *WFDy* coincides with the well founded semantics of GLPs [6] when the considered DyLP is a single program P, and with the well founded semantics of normal programs [11] when, furthermore, that single program has no negation in rule heads. Denoting by *WFG(P)* the well founded semantics of the generalized logic program P according to [6]:

Theorem 4. *Let P be a generalized program. Then WFG(P) = WFDy(P).*

Since *WFG(P)* coincides with the well founded semantics of [11] when P is a normal program (cf. [6]), it follows that *WFDy* coincides with the semantics of [11] whenever the sequences is made of a single normal logic program.

5.1 Brief Comparisons

Among the various semantics defined for sequences of LPs [1,2,5,9,14,15,18], *WFDy* shares a close relationship with the refined stable models semantics of [1], resembling that between stable and well founded semantics of normal programs.

Proposition 2. *Let M be any refined stable model of* $\oplus P_i$. *The well founded model* $WFDy(\oplus P_i)$ *is a subset of M. Moreover, if* $WFDy(\oplus P_i)$ *is a 2-valued interpretation, it coincides with the unique refined stable model of* $\oplus P_i$.

This property does not hold if, instead of the refined semantics, we consider any of the other semantics based on causal rejection [2,5,9,14,15]. This is so because these semantics are sometimes overly skeptical, in the sense that admit to many models, and thus have a smaller set of conclusions (in the intersection of all stable models). One particular case when this happens is when a sequence is updated with tautologies. Though, intuitively, updating our knowledge with a tautology should have no influence on the results, this is not the case in any of those semantics. For example, with all the cited semantics, updating the program $P_1 \oplus P_2$ of example 2 (which, according to all of them, has a single stable model containing *not see_stars*) with the program P_3 : $see\_stars \leftarrow see\_stars$ leads to two stable models: one in which *see_stars* is true and the other in which *see_stars* is false. Thus, this intuitively harmless update prevents *not see_stars* from being concluded. This is not the case with [1], nor with *WFDy*, which are both immune to tautologies. In this example, both conclude that *not see_stars* is true, before and after the update. For further details on this topic see [1].

Notably, all the attempts found in the literature [2, 3, 13] to define a well founded semantics for logic programs updates are overly skeptical as well. According to all the cited semantics, the well founded model of $P_1 \oplus P_2 \oplus P_3$ is the empty set, hence, unlike *WFDy*, they are not able to conclude that *not see_stars* is true. Though there it cannot be detailed here, other class of programs exist, besides the ones with tautologies, where the cited semantics bring more skeptical results then *WFDy*. Moreover the definition of all these semantics is based on a complex syntactical transformation of sequences into single programs, making it difficult to grasp what the declarative meaning of a sequence is.

6 Concluding Remarks

Guided by the needs of applications, it was our purpose in this paper to define a semantics for DyLPs fulfilling some specific requirements. Namely: a semantics whose computation has polynomial complexity; that is able to deal with contradictory programs, assigning them a non trivial meaning; that can be used to compute answers to queries without always visiting the whole knowledge base. With this in mind, we have defined a well founded semantics for DyLPs.

The defined semantics is a generalization of the well founded semantics of normal and. generalized logic programs, and it coincides with them when the DyLP consists of a single program. It has polynomial complexity and obeys relevance. Regarding the requirement of being able to deal with contradictions, lack of space prevented us from further elaborating on the properties of the semantics in these cases, and on how it can in general be used to detect contradictory literals and literals that depend on contradictions. We have, nonetheless, provided a complete characterization of the non contradictory cases.

We briefly compared the proposed semantic to the existent ones for DyLPs that are based on the causal rejection principle, and shown that it is a skeptical approximation of the refined stable model semantics for DyLPs, and less skeptical than all other existing well founded based semantics for updates. Comparisons to semantics of updates that are not based on the causal rejection principle are outside the scope of this paper. For an analysis of these semantics, and comparisons to the above ones see e.g. [14].

References

1. J. J. Alferes, F. Banti, A. Brogi, and J. A. Leite. Semantics for dynamic logic programming: a principled based approach. In V. Lifschitz and I. Niemelä, editors, *LPNMR-7,* volume 2923 of *LNAI,* pages 8–20. Springer, 2004.
2. J. J. Alferes, J. A. Leite, L. M. Pereira, H. Przymusinska, and T. C. Przymusinski. Dynamic updates of non-monotonic knowledge bases. *The Journal of Logic Programming,* 45(1–3) :43–70, September/October 2000.
3. J. J. Alferes, L. M. Pereira, H. Przymusinska, and T. Przymusinski. LUPS: A language for updating logic programs. *Artificial Intelligence,* 132(1 & 2), 2002.
4. K. R. Apt and R. N. Bol. Logic programming and negation: A survey. *The Journal of Logic Programming,* 19 & 20:9–72, May 1994.
5. F. Buccafurri, W. Faber, and N. Leone. Disjunctive logic programs with inheritance. In D. De Schreye, editor, *ICLP'99,* pages 79–93. MIT Press, 1999.
6. C. V. Damásio and L. M. Pereira. Default negation in the heads: why not? In R. Dyckhoff, H. Herre, and P. Schroeder-Heister, editors, *Int. Ws. Extensions of Logic Programming,* volume 1050 of *LNAI.* Springer, 1996.
7. C. V. Damásio and L. M. Pereira. A survey on paraconsistent semantics for extended logic programas. In D. M. Gabbay and Ph. Smets, editors, *Handbook of Defeasible Reasoning and Uncertainty Management Systems,* volume 2, pages 241–320. Kluwer, 1998.
8. J. Dix. A classification theory of semantics of normal logic programs II: Weak properties. *Fundamenta Mathematicae,* 22(3):257–288, 1995.
9. T. Eiter, M. Fink, G. Sabbatini, and H. Tompits. On properties of semantics based on causal rejection. *Theory and Practice of Logic Programming,* 2:711–767, November 2002.
10. A. Van Gelder. The alternating fixpoint of logic programs with negation. *Journal of Computer and System Sciences,* 1992.
11. A. Van Gelder, K. A. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM,* 38(3):620–650, 1991.
12. M. Gelfond and V. Lifschitz. Logic programs with classical negation. In Warren and Szeredi, editors, *7th ICLP,* pages 579–597. MIT Press, 1990.
13. J. A. Leite. Logic program updates. Master's thesis, Dept. de Informática, Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa, November 1997.

14. J. A. Leite. *Evolving Knowledge Bases,* volume 81 of *Frontiers in Artificial Intelligence and Applications.* IOS Press, December 2002.
15. J. A. Leite and L. M. Pereira. Iterated logic program updates. In J. Jaffar, editor, *JICSLP-98,* pages 265–278. MIT Press, 1998.
16. V. Lifschitz and T. Woo. Answer sets in general non-monotonic reasoning (preliminary report). In B. Nebel, C. Rich, and W. Swartout, editors, *KR-92.* Morgan-Kaufmann, 1992.
17. C. Sakama and K. Inoue. Updating extended logic programs through abduction. In M. Gelfond, N. Leone, and G. Pfeifer, editors, *LPNMR-99,* volume 1730 of *LNAI,* pages 147–161. Springer, 1999.
18. Y. Zhang and N. Y. Foo. Updating logic programs. In Henri Prade, editor, *ECAI-98,* pages 403–407. John Wiley & Sons, 1998.

A First Order Temporal Logic for Behavior Representation

Carlos Rossi, Manuel Enciso, and Ángel Mora*

E.T.S.I. Informática, Universidad de Málaga,
Campus de Teatinos, 29071 Málaga, Spain
rossi@ctima.uma.es

Abstract. An important problem for knowledge representation is the specification of the behavior of information systems. To solve this problem, different formal techniques have been used and many authors have endorsed the use of different sorts of logics.

This problem is even more important in software engineering, where the main modeling languages are defined with no formal semantics. For example UML 2.0 has been proposed using a textual semantics.

This work has a twofold objective: Firstly, we introduce a novel modal temporal logic called LNi^1, which is the natural extension to first order of LNint-e, presented in [17,7]. Secondly, we show the usefulness of our logic for solving an important and specific knowledge representation problem: Providing UML with a formal semantics (focusing on state machines). This way we want both to avoid the disadvantages of current *textual* UML semantics and to provide a formal basis for further verification and validation of UML models. Our new logic LNi^1 overcomes the two main limitations of LNint-e in the formalization of UML state machines: the use of parameters in the operations and the specification of the communications between objects.

Keywords: Interval temporal logic, knowledge representation, formal semantics, intelligent information systems, state machines.

1 Introduction

In the field of software engineering the need for a formalization to achieve precise unambiguous specifications is widely accepted. These specifications conform the previous and necessary step for formal property verification.

This formalization can be achieved using several techniques (logic, algebraic specifications, etc.). Chomicki and Saake [3] make a stand for logic in the following statement: *"Logic has simple, unambiguous syntax and semantics. It is thus ideally suited to the task of specifying information systems"*.

Temporal logic is a natural candidate for the formalization of dynamic behavior of software systems. This is supported by its use by well known authors as Pnueli [14] or Lamport [13] as for its recent use in CASE tools that include

* The authors are partially supported by CICYT project TIC2003-08687-C02-01.

formal methods in a software developing process [19,12]. In [3] the appropriateness of temporal logic for computing is commented. *"although temporal logic has been studied by logicians for a long time, its use in this area is new and leads to many interesting research problems".*

The main goal of this work is to deepen the use of temporal logic in knowledge representation, specifically, in this work we make two contributions:

- The first one is the development of a novel temporal first order logic of points and intervals. This modal logic, that we call LNi^1, is the extension of the propositional logic LNint-e [5,17]. LNi^1 is a many-sorted logic and all its domains are finite or countable.

  LNi^1 is characterized for combining expressions of points and intervals and the relative and absolute temporal approaches. We must remark the use in LNi^1 of temporal connectives based in precedence and posteriority concepts and its topological semantics [2].
- The second one proves the real applicability of our logic to a specific knowledge representation problem: The modeling of software system's behavior. In that respect we have focused on UML, the standard modeling language nowadays. More specifically, on their main behavioral model: state machines.[1]

  It is important to remark that we developed a previous formalization of state machines using the propositional logic LNint-e [17,7]. The first order logic developed in this work constitutes an important advance, as it allows the formalization of parameters in the actions, as well as the specification of the communications between objects. Both are basic aspects in the behavior of software systems that could not be dealt with in previous formalizations.

  In the literature it is possible to find several different formalizations of the state machines based in different techniques: graphs, automata, logic, Petri nets, model checking. A thorough study of these different approaches can be found in [17,20].

This work is structured as follows: In the next section we present the logic LNi^1 and we detail its syntax and semantics, and in section 3 we show the main novelties in our state machines formalization. The conclusions and future works sections can be found at the end.

2 LNi^1 Logic

In this work we present the logic LNi^1, a modal temporal logic that is a first order extension of the temporal propositional logic LNint-e, introduced in [17,5]), which uses $(\mathbb{Z}, \leq)$ as the flow of time.

2.1 Syntax of LNi^1

Alphabet. Before starting the composition of the alphabet, it is convenient to show the main ideas that guide the construction of our logic: We have decided to

[1] State machines were already a consolidated model before UML appeared [11].

combine points and intervals as temporal primitives. In addition, going further in the study of temporal ontologies (a detailed study can be found in [18]) we have reached the conclusion of the necessity of two kinds of interval expressions, namely :

– *Hereditary Expressions* [18]: When affirmed at an interval, are also true at the points of that interval. In LNi^1 this kind of expressions share a representation with point expressions.
– *Non-hereditary Expressions* [18]: When affirmed at an interval, are not true at the points of that interval. At this point we must remark that we distinguish between non-hereditary types and non-hereditary executions that represent each one of the concrete occurrences of a non-hereditary type. Each one of these executions will be distinguished by means of a label.

Before presenting the alphabet of LNi^1, we would like to state that it is defined taking into consideration the domains that we use when applying it to software modeling. These domains refer to classes and objects (in the usual sense of object oriented paradigm) and labels for non-hereditary executions. According to this, the alphabet of LNi^1 includes the following symbols:

– The set of classical connectives $\{\neg, \vee, \wedge, \rightarrow, \forall, \exists\}$ and the set of binary temporal connectives of points $\{\preccurlyeq, \not\succcurlyeq\}$.
– The symbols $\top$ and $\bot$, to denote truth and falsity, respectively.
– The sets $\mathcal{C}_c = \{c, c_1, .., c_n, ..\}$, $\mathcal{C}_o = \{o, o_1, .., o_n, ..\}$ and $\mathcal{C}_l = \{l, l_1, .., l_n, ..\}$ of symbols to denote class, object and label constants, respectively.
– The sets $\mathcal{V}_c = \{x, x_1, .., x_n, ..\}$, $\mathcal{V}_o = \{y, y_1, .., y_n, ..\}$ and $\mathcal{V}_l = \{z, z_1, .., z_n, ..\}$ of symbols to denote class, object and label variables, respectively.
– The sets $\mathcal{F}_c = \{f_c, f_{c1}, \ldots, f_{cn}, \ldots\}$, $\mathcal{F}_o = \{f_o, f_{o1}, \ldots, f_{on}, \ldots\}$ and $\mathcal{F}_l = \{f_l, f_{l1}, \ldots, f_{ln}, \ldots\}$ of symbols to denote class, object and label functions, respectively.
– The following sets of predicate symbols: $\mathcal{P}_p = \{P, Q, .., P_1, Q_1, .., P_n, Q_n, \ldots\}$ to denote point predicates and hereditary interval predicates; $\mathcal{P}_{nh} = \{\alpha, \beta, \ldots, \alpha_1, \beta_1, \ldots, \alpha_n, \beta_n, \ldots\}$ to denote non-hereditary interval predicates; and a set $\underline{\mathbb{Z}} = \{\underline{m} \mid m \in \mathbb{Z}\}$ to denote *dates,* i. e. to name known instants.[2]
– The symbols $\uparrow, \downarrow, \rightarrow$, [,] , (,) and ",".

The LNi^1 Language. At this point we present the LNi^1 language, firstly stating its terms and later its atoms and finally its well-formed formulas.

Terms.

1. The constants and variables are terms. Depending on the types considered in the alphabet we will set a difference between class terms, object terms and label terms. We use $\mathcal{T}_l$, $\mathcal{T}_o$ and $\mathcal{T}_c$ to denote the sets of label, object and class terms, respectively. We define $\mathcal{T} = \mathcal{T}_l \cup \mathcal{T}_o \cup \mathcal{T}_c$, the set of terms.

[2] Date predicates have arity 0.

2. Given a symbol of function $f \in \mathcal{F}_d$, with $d \in \{l, o, c\}$ of arity n and the terms $t_1, \ldots, t_n \in \mathcal{T}$, then $f(t_1, \ldots, t_n) \in \mathcal{T}_d$.
   Each n-ary function would have a signature belonging to $\mathcal{S}^n$, where $\mathcal{S} \in \{\mathcal{T}_l, \mathcal{T}_o, \mathcal{T}_c\}$. For example, given a binary object function f_o (with signature $\mathcal{T}_o \times \mathcal{T}_c$) and two terms $t_1 \in \mathcal{T}_o$ and $t_2 \in \mathcal{T}_c$, then $f_o(t_1, t_2)$ is an object term.
3. There are no more terms than those built as stated in items 1 and 2.

Atoms. We will set three types of atoms:

1. *Point or Hereditary Atoms:* Are those of the form $P(t_1, \ldots, t_n)$, where P is a symbol of point or hereditary n-ary predicate and $t_1, \ldots, t_n \in \mathcal{T}$. We denote Ω_p as the set of point atoms. Each n-ary point or hereditary predicate will have a signature belonging to $\mathcal{S}^n$, where $\mathcal{S} \in \{\mathcal{T}_l, \mathcal{T}_o, \mathcal{T}_c\}$.
2. *Date Atoms:* If $\underline{m} \in \mathbb{Z}$ then $\underline{m}$ is a date atom.
3. *Atoms Derived from Non-hereditary Expressions:* The following type of atoms implements some of the essential ideas in the construction of LNi^1. As previously stated, our logic has a great expressive power as it combines points and interval expressions. Nevertheless, we thought it appropriate to characterize interval expressions by means of expressions of points, as doing so, we obtain some advantages from the point of view of complexity: on the one hand, all the logic expressions become point expressions, and these use simpler connectives; on the other hand, we avoid the ambiguous concept of the current interval that the interval modal logics impose.

   This characterization is direct in hereditary interval expressions, as, by definition, they are "inherited" by the points that are part of the interval. Nevertheless, this is not the case with non-hereditary interval expressions. Thus, we decided to characterize these expressions by their start and end instants, and their course.

   On the other hand, we have to keep in mind that regarding non-hereditary expressions in LNi^1 we set apart non-hereditary types and non-hereditary executions. In LNi^1 the non-hereditary types are represented by means of non-hereditary predicates and the labels that represent the executions of this type are included as an extra argument in the predicate and is shown as a subindex. For example, let us consider a non-hereditary type "Calculate Total" and that it has two arguments. The representation in LNi^1 will be by means of a non-hereditary predicate CalcularTotal. If we want to represent a specific execution of that non-hereditary type to two objects o_1 and o_2 and we assume that the label associated to that execution is l, we would write in LNi^1 $\mathsf{CalcularTotal}_l(o_1, o_2)$.

   All this argumentation can be summed up as follows: for each symbol of non-hereditary n-ary predicate $\alpha \in \mathcal{P}_{nh}$, $l \in \mathcal{T}_l$ and $t_1, \ldots, t_n \in \mathcal{T}$, then three kinds of associated (point) atoms are defined:
   - $\uparrow \alpha_l(t_1, .., t_n)$ that represents the start instant of the execution $\alpha_l(t_1, .., t_n)$
   - $\downarrow \alpha_l(t_1, .., t_n)$ that represents the end instant of the execution $\alpha_l(t_1, .., t_n)$
   - $\overrightarrow{\alpha_l(t_1, .., t_n)}$ that represents an instant during the course of the execution $\alpha_l(t_1, .., t_n)$

Every n-ary non-hereditary predicate will have a signature belonging to $\mathcal{S}^n$, where $\mathcal{S} \in \{\mathcal{T}_l, \mathcal{T}_o, \mathcal{T}_c\}$.

Well-Formed Formulas (wffs): We already have all the necessary elements to present well-formed formulas included in the language of LNi^1:

1. Atoms, $\top$ and $\perp$ are *wffs*
2. If A and B are *wffs*, $\neg A$, $A \vee B$, $A \wedge B$, $A \preccurlyeq B$ and $A \succcurlyeq B$ are *wffs* too
3. If A is a *wff* and x a variable, $\forall x A$ and $\exists x A$ are *wffs* too
4. There are no more *wffs* than the ones formed according to these points.

The readings of all these *wffs* are well-known except those of the temporal connectives of LNi^1, $\preccurlyeq$ and $\succcurlyeq$, that include the temporal relationships of precedence and posteriority: These reading will be completely detailed in the following section, devoted to the semantics of LNi^1.

2.2 Semantics

LNi^1 is a many-sorted logic with the following domains: $\mathcal{C}$ for classes, $\mathcal{O}$ for objects and $\mathcal{L}$ for labels. The first two are finite, and the referent to labels will be countable.[3] We denote $\mathcal{D} = \{\mathcal{C}, \mathcal{O}, \mathcal{L}\}$.

From these domains we can present the concept of interpretation for LNi^1:

Definition 1. *A interpretation for LNi^1 is a tuple $(\mathcal{C}, \mathcal{O}, \mathcal{L}, I)$, where $\mathcal{C}$, $\mathcal{O}$ and $\mathcal{L}$ are non empty sets that represent domains of classes, objects and labels, respectively, and I is an application that associates:*

- To each class constant symbol c class an element $I(c) \in \mathcal{C}$. The interpretation of object and label constants is analogously constructed.
- To each n-ary class function symbol f_c an n-ary function over $\mathcal{D}^n$, that is, $I(f_c) : \mathcal{D}^n \rightarrow \mathcal{C}$. The interpretation of object and label functions is analogously constructed.
- To each symbol of n-ary point predicate P an n-ary relation over $\mathcal{D}^n$, that is, $I(P) \subseteq \mathcal{D}^n$
- To each symbol of n-ary non-hereditary predicate α an n-ary relation on $\mathcal{L} \times \mathcal{D}^n$, that is, $I(\alpha) \subseteq \mathcal{L} \times \mathcal{D}^n$

Given an interpretation $(\mathcal{C}, \mathcal{O}, \mathcal{L}, I)$, the variables have the expected meaning, that is, they represent arbitrary elements of the corresponding domain.

Before defining the concept of temporal interpretation we need to define some other previous concepts:

- Let $nhEx = \{\alpha_l(t_1, \ldots, t_n) \mid \alpha \in \mathcal{P}_{nh}, l \in \mathcal{T}_l, t_1, \ldots, t_n \in \mathcal{T}\}$ be the set of all non-hereditary executions. The terms $t_1, \ldots, t_n$ must be of the adequate type to the signature of α.

[3] As a matter of fact, except in modeling of objects of permanent existence (that is, those that are carrying out actions for an infinite time), a finite domain suffices.

- We denote $Int(\mathbb{Z}) = \{[i_1, i_2] \mid i_1, i_2 \in \mathbb{Z}, i_1 < i_2\}$ as the set of closed finite intervals of $\mathbb{Z}$ with different start and end points.
- Let $\Omega_{der} = \{\uparrow \alpha_l(t_1, .., t_n), \overrightarrow{\alpha_l(t_1, .., t_n)}, \downarrow \alpha_l(t_1, .., t_n) \mid \alpha \in \mathcal{P}_{nh}, l \in \mathcal{T}_l, t_1, .., t_n \in \mathcal{T}\}$ be the set of points atoms derived from the non hereditary ones.
- We denote $\Omega_p^{nh} = \Omega_p \cup \mathbb{Z} \cup \Omega_{der} \cup \top \cup \bot$

Definition 2. *A* **temporal interpretation** *for LNi^1 is a pair of functions $\mathcal{TI} = \langle H_{exec}, h\rangle$, where:*

- *H_{exec}: $nhEx \rightarrow Int(\mathbb{Z})$ associates each non-hereditary execution $\alpha_l(t_1, .., t_n)$ with the only interval where $\alpha_l(t_1, \ldots, t_n)$ holds.*

  We must remark that the semantics of LNi^1 avoids the important restriction of LNint-e that imposed that all the non-hereditary executions of the same non-hereditary class had to have the same duration [17].
- *h: $\Omega_p^{nh} \longrightarrow 2^{\mathbb{Z}}$ associates each atom of Ω_p^{nh} with a subset $\mathbb{Z}$ that satisfies the following conditions:*
  1. *$h(\bot) = \emptyset$; $h(\top) = \mathbb{Z}$ and $h(\underline{m}) = \{t\}$, for all $\underline{m} \in \underline{\mathbb{Z}}$*
  2. *For all $\alpha_l(t_1, .., t_n) \in EventExecs$, if $H_{exec}(\alpha_l(t_1, .., t_n)) = [i_1, i_2]$, then: $h(\uparrow\alpha_l(t_1, .., t_n)) = \{i_1\}$, $h(\overrightarrow{\alpha_l(t_1, .., t_n)}) = (i_1, i_2)$ and $h(\downarrow\alpha_l(t_1, .., t_n)) = \{i_2\}$.*

Naturally, it is now necessary to extend the concept of temporal interpretation to any *wff* of LNi^1. We will present this extension exclusively for the future fragment of LNi^1 (for the past fragment it is symmetrically extended). The extension is based in the key concept of our topological semantics, denoted m_{tA}^+, that represents the following idea: If A is a wff of LNi^1 and $t \in \mathbb{Z}$, we define: $m_{tA}^+ = min\{t' \in \mathbb{Z} \mid t' > t$ and A is true at $t'\}$

In other words, m_{tA}^+ is the first instant after t in which A will be true. From these concepts we define the extension of a temporal interpretation $\mathcal{TI} = \langle H_{exec}, h\rangle$, that actually only requires the extension of the function h to any *wff*. This extension becomes the usual form for boolean connectives. The extension for the future temporal connective considering that A, B are *wff* of LNi^1, is

$$h(A \preccurlyeq B) = \{t \in \mathbb{Z} \mid m_{tA}^+ < +\infty \text{ and } m_{tA}^+ \leq m_{tB}^+\}$$

where m_{tA}^+ is formally defined as follows:

Definition 3. *Let A be a wff of LNi^1 and $t \in \mathbb{Z}$, we define* $m_{tA}^+ = \min\left((t, +\infty) \cap h(A)\right)$.

We agree that $min\, \emptyset = +\infty$.

Therefore, the meaning of the temporal connective $\preccurlyeq$ is the following:

$A \preccurlyeq B$ is read as *sometime in the future A, and the next occurrence of A will be before or simultaneous to the next occurrence of B.*

The concepts of validity and satisfiability are defined in the usual manner.

2.3 Temporal Relations in LNi^1

In this section we introduce some defined connectives needed in our application.

Point Relations. The system $\{\preccurlyeq, \succcurlyeq\}$ has fully expressive power regarding point expressions [2]. So, we may define[4] other well known temporal connectives: $\Diamond$ (sometimes in the future), $\Box$ (always in the future), $\oplus$ (next) and $\ominus$ (prior).

Here, we present new point connectives used to formalize UML state machines:

- $A \approx^+ B$ is read as *sometime in the future, A and B will occur and the next occurrences of A and B will be simultaneous.* The definition in LNi^1 is $A \approx^+ B \overset{\text{def}}{=} A \preccurlyeq B \wedge B \preccurlyeq A$.
- *inst(A)* is read as *at the current instant A is true in the form of an isolated point.* The definition is $inst(A) \overset{\text{def}}{=} \ominus\neg A \wedge A \wedge \neg \oplus A$.
- $Int^+(A)$ is read as *in the future A will occur, and the next time that A occurs, A holds at a finite closed interval.* The definition is $Int^+(A) \overset{\text{def}}{=} (A \approx^+ (A \wedge \oplus A \wedge \Diamond\neg A)) \wedge (\oplus A \rightarrow \neg A)$.

Interval Relations. The expressive power of LNi^1 regarding non-hereditary expressions is at least equal to that of Allen [1] and Halpern and Shoham [10]. Then, in our logic it is possible to define the standard temporal relations between non hereditary and/or hereditary expressions [16]. We present here three interval connectives needed to formalize UML state machines whose reading is the usual in [1,10]:

- $ab^+(\alpha_l, \beta_{l'})$ is the *abutment relation between two event executions.* The definition in LNi^1 is
  $ab^+(\alpha_l, \beta_{l'}) =_{def} \Diamond \uparrow \alpha_l \wedge \downarrow \alpha_l \approx^+ \uparrow \beta_{l'} \wedge (\forall n, n' \uparrow \alpha_l \preccurlyeq \uparrow \alpha_n \wedge \uparrow \beta_{l'} \preccurlyeq \uparrow \beta_{n'})$
- $ab^+_{nh\text{-}h}(\alpha_l, A)$ is the *abutment relation between the non-hereditary execution α_l and the hereditary expression A.* Its definition is
  $ab^+_{nh-h}(\alpha_l, A) =_{def} Int^+(A) \wedge \downarrow \alpha_l \approx^+ \uparrow A \wedge (\forall n \uparrow \alpha_l \preccurlyeq \uparrow \alpha_n)$
- $\uparrow sec^+(\alpha_l, \beta_{l'}, \delta_{l''})$ represents *the starting point of a sequence of three non-hereditary executions.*
  $\uparrow sec^+(\alpha_l, \beta_{l'}, \gamma_{l''}) =_{def} \uparrow \alpha_l \wedge \downarrow \alpha_l \approx^+ \uparrow \beta_l \wedge \downarrow \beta_{l'} \approx^+ \uparrow \gamma_{l''} \wedge (\forall n, n' \uparrow \beta_{l'} \preccurlyeq \uparrow \beta_n \wedge \uparrow \gamma_{l''} \preccurlyeq \uparrow \gamma_{n'})$

3 UML State Machines in LNi^1

As we mentioned in the introduction, the main objective of this work is to illustrate the usefulness of temporal logic in knowledge representation and, specifically, in software engineering.

The specific problem that we try to solve is the lack of formalization in UML which makes it difficult to verify the models. The authors of UML has

[4] In [2] we introduce these definitions formally.

approached this by introducing the OCL language, although it is only used to formalize well-formedness rules whereas the semantics is described textually. As a matter of fact, the formalization of UML semantics is one of the most active research tasks in software engineering, and specifically the formalization of the UML state machines. This is reflected in several works that have been analyzed in detail in [17].

In this section we show how LNi^1 improves previous works devoted to providing a formal semantics to UML state machines[5]. Due to space limitations, we focussed only on the novelties of this work. The unfamiliar reader may get a deep description of UML in [15] and a full specification of UML state machines using the propositional temporal logic LNint-e in [17].

The main contribution of this work is the formalization of the use of parameters in the different operations of the state machine, which is an essential aspect in any software system and in particular in the communication among objects.

State machines operations need a time-interval to be executed. So, they are represented by non-hereditary atoms, since we must represent complete executions of operations. Now we present different kinds of operations:

1. *Actions* are operations whose executions cannot be aborted. To specify an action act, with parameters $\mathsf{o}_1, \ldots, \mathsf{o}_n$, each one of them of the $\mathsf{c}_1, \ldots, \mathsf{c}_n$, class respectively, a predicate act will be included in the language. That predicate will have as many arguments as the state machine action has.
   If the argument corresponds to an object, we must also specify the class to which it belongs. Therefore, the representation of the action $\mathsf{act}(\mathsf{o}_1, \ldots, \mathsf{o}_n)$ will be $\mathsf{act}_l(\mathsf{o}_1 : \mathsf{c}_1, \ldots, \mathsf{o}_n : \mathsf{c}_n)$. In the specification an ownership relation between each object and its class must be included.
2. *Activities* can be aborted by an exiting transition and so the duration of each execution may be different. LNi^1 overcomes an important problem in our previous formalization based on LNint-e, in which it was required that all executions on an non-hereditary type must have the same duration and a connective, named *ExecAbort,* represented an activity being aborted. This mechanism is no longer necessary in LNi^1.
3. *Send events* are another option among the actions of a transition or state and they represent the communication between objects. Their formalization (impossible in LNint-e and in other works based on propositional logic) is another main contribution of LNi^1. Send events are actions and they are also specified by non hereditary predicates.

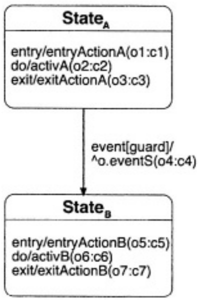
   Generically, we consider a predicate $\mathsf{send} - \mathsf{ev}$ to specify the sending of a call event ev. That predicate will have as many arguments as the event sent has and one more to represent the target object (and its class). Therefore, the representation of the sending of event $\mathsf{ev}(\mathsf{o}_1, \ldots, \mathsf{o}_n)$ to an object o of the c class will be $\mathsf{send} - \mathsf{ev}_l(\mathsf{o} : \mathsf{c}; \mathsf{o}_1 : \mathsf{c}_1, \ldots, \mathsf{o}_n : \mathsf{c}_n)$, where l is an unused label, and $\mathsf{c}_1, \ldots, \mathsf{c}_n$ are the classes of $\mathsf{o}_1, \ldots, \mathsf{o}_n$, respectively.

[5] We have to remark that our formalization respects the execution model of state machines proposed in UML, based in the run-to-completion assumption [15].

In our approach, the start instant of the event corresponds with the instant the event is sent, and the final instant with the end of execution of the invoked operation of the target object. This characteristic allows us to specify the behavior of the synchronous communication between objects. As indicated in the UML semantics [15], the object that sends the event is blocked until the invoked operation is completed.

The formalization of rest of the elements of the UML state machines is a natural extension of that proposed for LNint-e [17]. In the formalization of those elements the LNi^1 temporal connectives are intensively used. As an example, we show the formalization of the generic transition connective[6] that represents the situation depicted in the figure below.

$$\begin{aligned}
&\mathsf{Transition}(\mathsf{State_A}, \mathsf{ActivA}_{\mathsf{I_2}}(\mathsf{o2}:\mathsf{c2}), \mathsf{ExitActionA}_{\mathsf{I_3}}(\mathsf{o3}:\mathsf{c3}),\\
&\quad \mathsf{Event}, \mathsf{Guard}, \mathsf{send}-\mathsf{eventS}_{\mathsf{I}}(\mathsf{o}:\mathsf{c};\mathsf{o4}:\mathsf{c4}),\\
&\quad \mathsf{State_B}, \mathsf{EntryActionB}_{\mathsf{I_5}}(\mathsf{o5}:\mathsf{c5}), \mathsf{ActivB}_{\mathsf{I_6}}(\mathsf{o6}:\mathsf{c6})) =_{def}\\
&\Box\Big(\mathsf{State_A} \wedge \mathsf{Event} \wedge \mathsf{Guard} \rightarrow \Big[\oplus\neg\mathsf{State_A} \;\wedge\\
&\qquad \textit{//if the transition is fired, the state ends}\\
&\qquad \Big[\overrightarrow{\mathsf{ActivA}_{\mathsf{I_2}}(\mathsf{o2}:\mathsf{c2})} \rightarrow \oplus\downarrow \mathsf{ActivA}_{\mathsf{I_2}}(\mathsf{o2}:\mathsf{c2})\Big] \wedge\\
&\qquad \textit{//if the activity is in process, it is aborted}\\
&\qquad \oplus \uparrow\!\mathsf{sec}^{+}(\mathsf{ExitAction}_{\mathsf{I_3}}(\mathsf{o3}:\mathsf{c3}),\\
&\qquad \mathsf{send}-\mathsf{eventS}_{\mathsf{I}}(\mathsf{o}:\mathsf{c};\mathsf{o4}:\mathsf{c4}), \mathsf{EntryActionB}_{\mathsf{I_5}}(\mathsf{o5}:\mathsf{c5})) \wedge\\
&\qquad \textit{//exit, event, and entry actions are executed}\\
&\qquad \mathsf{ab}^{+}_{\mathsf{nh-h}}(\mathsf{EntryActionB}_{\mathsf{I_5}}(\mathsf{o5}:\mathsf{c5}), \mathsf{State_B}) \wedge\\
&\qquad \textit{//target state is started}\\
&\qquad \mathsf{ab}^{+}(\mathsf{EntryActionB}_{\mathsf{I_5}}(\mathsf{o5}:\mathsf{c5}), \mathsf{ActivB}_{\mathsf{I_6}}(\mathsf{o6}:\mathsf{c6}))\Big]\Big)\\
&\textit{// and the execution of the target state activity is started}
\end{aligned}$$



4 Conclusions and Future Work

In this work we present LNi^1, a first-order temporal logic that combines points and intervals and the absolute and relative approaches. It is a many-sorted logic, whose domains are finite or countable. We have demonstrated the public usefulness of LNi^1 for an open problem, as the one of providing a formal semantics to state machines, the main behavior model of UML, the standard modeling language for software systems.

As future work, an important aspect in logic is the availability of reasoning methods. In that sense, our research group has achieved great advances in the development of automated theorem provers for temporal logics [6, 4], that will

[6] Transitions are represented by means of connectives that relate all the elements that can participate in a transition, namely: events, guards and actions, as well as the source and target states.

help to develop reasoning methods for LNi^1. In this line of thought, we have defined a normal form that is independent from the deduction method for LNint-e [8]. We will take into consideration some other recent works for first order temporal logics [9]. Our intention is to integrate deduction methods into a CASE tool of which we already have a prototype. It allows us to edit statecharts and to automatically generate the associated LNi^1 formulae.

References

1. J. F. Allen. Towards a general theory of action and time. *Artificial Intelligence,* 23(2):123–154, 1984.
2. A. Burrieza and I. P. de Guzmán. A new algebraic semantic approach and some adequate connectives for computation with temporal logic over discrete time. *Journal of Applied non Classical Logic,* 2, 1992.
3. J. Chomicki and G. Saake, editors. *Logics for database and information systems.* Kluwer Academic Publishers, 1998.
4. P. Cordero, M. Enciso, and I. P. de Guzmán. Bases for closed sets of implicants and implicates in temporal logic. *Acta Informatica,* 38:599–619, 2002.
5. I. Pérez de Guzmán and C. Rossi. LNint: a temporal logic that combines points and intervals and the absolute and relative approaches. *Journal of the IGPL,* 3(5), 1995.
6. M. Enciso, I. P. de Guzmán, and C. Rossi. Temporal reasoning over linear discrete time. In *Logics in Artificial intelligence,* number 1126 in Lecture Notes in Artificial Intelligence, pages 303–319. Springer Verlag, 1996.
7. M. Enciso, I. P. de Guzmán, and C. Rossi. Using temporal logic to represent dynamic behaviour of UML statecharts. In J. Hernández and A. Moreira, editors, *ECOOP 2002 Workshop on Integration and Transformation of UML Models,* volume 2548 of *Lecture Notes in Computer Science.* Springer-Verlag, 2002.
8. M. Enciso, I. P. de Guzmán, and C. Rossi. Una forma normal temporal independiente del método de deducción. *Inteligencia Artificial,* 8(23):55–74, 2004.
9. D. P. Guelev. A complete proof system for first-order interval temporal logic with projection. *Journal of Logic and Computation,* 14(2):215–249, 2004.
10. J. Halpern and Y. Shoham. A propositional modal logic of time intervals. *Journal of the ACM,* 38(4):935–962, 1991.
11. D. Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming,* 5:231–274, 1987.
12. A. Knapp, S. Merz, and C. Rauh. Model Checking Timed UML State Machines and Collaborations. In *Proc. 7th Int. Symp. Formal Techniques in Real-Time, and Fault Tolerant Systems,* volume 2469 of *Lecture Notes in Computer Science,* pages 395–416. Springer, 2002.
13. L. Lamport. The temporal logic of actions. *ACM Transactions on Programming Languages and Systems,* 16(3):872–923, 1994.
14. Z. Manna and A. Pnueli. *Temporal verification of reactive systems: Safety.* Springer-Verlag, 1995.
15. Object Management Group. *UML 2.0 Specification,* 2004. `http://www.omg.org/uml`.
16. C. Rossi. *Lógica Temporal de Intervalos. Formalización de Diagramas de Estados.* PhD thesis, Universidad de Málaga, Spain, 2001.
17. C. Rossi, M. Enciso, and I. P. de Guzmán. Formalization of UML state machines using temporal logic. *Journal on Software and Systems Modeling,* 3(1):31–54, 2004.

18. Y. Shoham. *Reasoning about Change. Time and Causation from the Standpoint of Artificial Intelligence.* MIT Press, 1988.
19. O. Slotosch. Overview over the project QUEST. In *Applied Formal Methods. Proceedings of FM-Trends 98, Boppard, Germany,* number 1641 in Lecture Notes in Computer Science, pages 346–350. Springer Verlag, 1999.
20. M. von der Beeck. A comparison of statechart variants. In *Formal Techniques in Real-Time and Fault-Tolerant Systems,* number 863 in Lecture Notes in Computer Science, pages 128–148. Springer-Verlag, 1994.

Improved Tupling for Optimizing Multi-paradigm Declarative Programs*

Soledad González and Ginés Moreno

Department of Computer Science, University of Castilla-La Mancha,
02071 Albacete, Spain
{sgonzalez, gmoreno}@info-ab.uclm.es

Abstract. This paper investigates the optimization by fold/unfold of declarative programs that integrate the best features from both functional and logic programming. Transformation sequences are guided by a mixed strategy which, in three low-level transformation phases, successfully combines two well-known heuristics -composition and tupling-, thus avoiding the construction of intermediate data structures and redundant sub-computations. In particular, whereas composition is able to produce a single function definition for some nested (composed) functions, the tupling method merges non-nested functions calls into a new function definition called *eureka*. We solve the non trivial problem of discovering the set of calls to be tupled in an incremental way, i.e. by chaining different *eureka* definitions where only non-nested calls sharing common variables are taken into account. Moreover, by appropriately combining both strategies, together with a simplification pre-process based on a kind of normalization, we automatically optimize a wide range of programs (with nested and/or non-nested function calls) at a very low cost.

1 Introduction

Functional logic programming languages combine the operational methods and advantages of the most important declarative programming paradigms, namely functional and logic programming. The operational principle of such modern multi-paradigm declarative languages is usually based on *narrowing*. A *narrowing step* instantiates variables in an expression and applies a reduction step to a redex of the instantiated expression. Needed narrowing is the currently best narrowing strategy for functional logic programs due to its optimality properties w.r.t. the length of derivations and the number of computed solutions [3], and it can be efficiently implemented by pattern matching and unification.

The fold/unfold transformation approach was first introduced in [5] to optimize functional programs and then used for logic programs [10]. A transformation methodology for lazy functional logic programs was presented in [2], where we also introduced an implemented prototype (the SYNTH tool) which has been successfully tested with several applications in the field of Artificial Intelligence. The

* This work was partially supported by CICYT under grant TIC 2001-2705-C03-03.

also called "rules+strategies" approach is commonly based on the construction, by means of a *strategy,* of a sequence of equivalent programs each obtained from the preceding ones by using an *elementary* transformation rule. The essential rules are *folding* and *unfolding,* i.e., contraction and expansion of subexpressions of a program using the definitions of this program or of a preceding one [9]. Although they are difficult to automate, there exists a large class of program optimizations such as composition and tupling (which is able to obtain even super-linear speedups), that can be achieved by fold/unfold transformations.

Composition essentially consists of the merging of nested function calls while the tupling strategy merges separate (non-nested) function calls with some common (variables) arguments into a single call to a (possibly new) recursive function which returns a tuple of the results of the separate calls, thus avoiding either multiple accesses to the same data structures or common subcomputations. In [1] and [8] we have investigated automatic strategies for performing composition and tupling, respectively, in a multi-paradigm declarative (functional-logic) setting. Moreover, as we show in this paper, by simplifying (normalizing) rules before applying composition and/or tupling, we can increase the efficiency and the applicability scope of the transformations. Since both strategies can be seen as antagonic in some senses, an important goal of the present work is to build a mixed heuristic that exploits the complementary effects of both methods in order to achieve better results (i.e., more general and powerful optimizations).

However, a weak point of these techniques is the achievement of the appropriate set of new *(eureka)* definitions which make it possible the optimizations to be pursued [5,10,9]. In contrast with prior non-automatic and rather complicated approaches (tupling has only been semi-automated to some -pure functional- extent [6,9]), this work starts from a previous (more realistic and simpler) method presented in [8], where we described the internal structure of tupling. Moreover, in the present approach we highly refine that algorithm by naturally embedding into it some simplification pre-processess, incremental capabilities, and more effective tests for termination. In particular, by performing recursive calls to the general tupling algorithm, we obtain more and more refined new eureka definitions until reaching the intended optimization, when possible.

The structure of the paper is as follows. After recalling some basic definitions, we introduce the basic transformation rules and illustrate its use in Section 2. The next section describes the different transformation phases that constitute the core of our composition and tupling algorithms. Section 4 proposes some simplification procedures based on normalization and composition, and shows that the three low-level transformation phases of tupling can be chained in an incremental way in order to reinforce its respective strengths. Finally, Section 5 concludes. More details can be found in [7].

2 Fold/Unfold Transformation Sequences

In this work we consider a *signature* Σ partitioned into a set $\mathcal{C}$ of *constructors* and a set $\mathcal{F}$ of *defined* functions. The set of *constructor terms* (with *variables*) is obtained by using symbols from $\mathcal{C}$ (and a set of variables $\mathcal{X}$). The set of variables occurring in a term t is denoted by $\mathcal{V}ar(t)$. We write $\overline{o_n}$ for the *list* of objects $o_1, \ldots, o_n$. A *pattern* is a term of the form $f(\overline{d_n})$ where $f/n \in \mathcal{F}$ and $d_1, \ldots, d_n$ are constructor terms (with variables). A term is *linear* if it does not contain multiple occurrences of one variable. $t|_p$ denotes the *subterm* of t at a given position p, and $t[s]_p$ denotes the result of *replacing the subterm* $t|_p$ by the term s. We denote by $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ the *substitution* σ with $\sigma(x_i) = t_i$ for $i = 1, \ldots, n$ (with $x_i \neq x_j$ if $i \neq j$), and $\sigma(x) = x$ for all other variables x. The application of a substitution σ to an expression e is denoted by $\sigma(e)$.

A set of rewrite rules $l \to r$ such that $l \notin \mathcal{X}$, and $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l)$ is called a *term rewriting system* (TRS). The terms l and r are called the *left-hand side* (lhs) and the *right-hand side* (rhs) of the rule, respectively. A TRS $\mathcal{R}$ is left-linear if l is linear for all $l \to r \in \mathcal{R}$. A TRS is constructor–based (CB) if each left-hand side is a pattern. In the remainder of this paper, a functional logic *program* is a left-linear CB-TRS without overlapping rules (i.e. the lhs's of two different program rules do not unify). A *rewrite step* is an application of a rewrite rule to a term, i.e., $t \to_{p,R} s$ if there exists a position p in t, a rewrite rule $R = (l \to r)$ and a substitution σ with $t|_p = \sigma(l)$ and $s = t[\sigma(r)]_p$. The operational semantics of modern integrated languages is usually based on *(needed) narrowing,* a combination of variable instantiation and reduction. Formally, $s \leadsto_{p,R,\sigma} t$ is a *narrowing step* if p is a non-variable position in s and $\sigma(s) \to_{p,R} t$. We denote by $t_0 \leadsto^*_\sigma t_n$ a sequence of narrowing steps $t_0 \leadsto_{\sigma_1} \ldots \leadsto_{\sigma_n} t_n$ with $\sigma = \sigma_n \circ \cdots \circ \sigma_1$.

We present now our program transformation method based on the fold/unfold methodology. First, we recall from [2] the set of (strong) correct/complete transformation rules that constitute the core of our transformation system. Basically, we start with an initial declarative program, $\mathcal{R}_0$, and construct a *transformation sequence* $\mathcal{R}_0, \ldots, \mathcal{R}_n$, $n > 0$, by applying the following transformation rules:

Definition Introduction: we may get program $\mathcal{R}_{k+1}$ by adding to $\mathcal{R}_k$ a new rule (called "definition rule" or "eureka") of the form $f(\overline{x_m}) \to r$, where $\mathcal{V}ar(r) = \{\overline{x_m}\}$ and f is a *new* function symbol not occurring in $\mathcal{R}_0, \ldots, \mathcal{R}_k$.

Unfolding: let $(l \to r) \in \mathcal{R}_k$ then, we may get program $\mathcal{R}_{k+1}$ as follows: $\mathcal{R}_{k+1} = \mathcal{R}_k \setminus \{l \to r\} \cup \{\sigma(l) \to r' \mid r \leadsto_\sigma r' \text{ in } \mathcal{R}_k\}$. We call normalizing step to a rewriting-based unfolding step (thus implying that $\sigma(l) \equiv l$).

Folding: let $(l \to r) \in \mathcal{R}_k$ be a non eureka rule[1] $(l' \to r') \in \mathcal{R}_j$, $0 \leq j \leq k$, an eureka rule, and p a position of r such that $r|_p = \sigma(r')$; then, we may get program $\mathcal{R}_{k+1}$ as follows: $\mathcal{R}_{k+1} = (\mathcal{R}_k \setminus \{l \to r\}) \cup \{l \to r[\sigma(l')]_p\}$.

Example 1. In order to optimize a program, the three rules above should be applied according to some appropriate *strategy* as it is the case of composition [5]. The following original program defines typical operations for computing the length and the sum of the elements of a list, and for generating a (descending) list of consecutive natural numbers:

[1] An eureka (or definition rule) maintains its status only as long as it remains unchanged, i.e., once it is transformed it is not considered an *eureka* rule anymore.

(R_1) : $\texttt{len([ ])} \rightarrow 0$ (R_2) : $\texttt{len([H|T])} \rightarrow \texttt{s(len(T))}$
(R_3) : $\texttt{sum([ ])} \rightarrow 0$ (R_4) : $\texttt{sum([H|T])} \rightarrow \texttt{H} + \texttt{sum(T)}$
(R_5) : $\texttt{gen(0)} \rightarrow [\,]$ (R_6) : $\texttt{gen(s(X))} \rightarrow \texttt{[s(X)|gen(X)]}$

If we want to add a new rule for obtaining the sum of the firsts X natural numbers, a naive possibility can be the following one (R_7) : $\texttt{sum\_from(X)} \rightarrow \texttt{sum(gen(X))}$. We can optimize this definition by avoiding the generation and subsequent traversal of the intermediate list by applying the composition strategy as follows:

1. Definition introduction: (R_8) : $\texttt{new}_1\texttt{(X)} \rightarrow \texttt{sum(gen(X))}$
2. Unfold rule R_8:
   (R_9) : $\texttt{new}_1\texttt{(0)} \rightarrow \texttt{sum([ ])}$ (R_{10}) : $\texttt{new}_1\texttt{(s(X))} \rightarrow \texttt{sum([s(X)|gen(X)])}$
3. Normalize rules R_9 and R_{10}:
   (R_{11}) : $\texttt{new}_1\texttt{(0)} \rightarrow 0$ (R_{12}) : $\texttt{new}_1\texttt{(s(X))} \rightarrow \texttt{s(X)} + \texttt{sum(gen(X))}$
4. Fold R_{12} using R_8: (R_{13}) : $\texttt{new}_1\texttt{(s(X))} \rightarrow \texttt{s(X)} + \texttt{new}_1\texttt{(X)}$
5. And finally, fold R_7 w.r.t R_8: (R_{14}) : $\texttt{sum\_from(X)} \rightarrow \texttt{new}_1\texttt{(X)}$

And now, the enhanced definition for sum_from corresponds to rule R_{14} toghether with the definition of the auxiliary symbol $\texttt{new}_1$ (rules R_{11} and R_{13}).

However, for the case of tupling we need to extend the core of our transformation system with an abstraction rule that allows us to flatten nested function calls or to pack them into a tuple as follows[2]:

Abstraction: let $R = (l \rightarrow r) \in \mathcal{R}_k$ be a rule and let $\overline{P_j}$ be sequences of disjoint positions in r such that $r|_p = e_i$ for all p in P_i, $i = 1, \ldots, j$, i.e., $r = r[\overline{e_j}]_{\overline{P_j}}$; then, we may get program $\mathcal{R}_{k+1}$ as follows: $\mathcal{R}_{k+1} = (\mathcal{R}_k \setminus R) \cup \{l \rightarrow r[\overline{z_j}]_{\overline{P_j}}$ *where* $\langle z_1, \ldots, z_j \rangle = \langle e_1, \ldots, e_j \rangle\}$, such that $\overline{z_j}$ are fresh variables, and local declarations are expressed with the typical *where* contruction of functional programming [5,4,9].

Example 2. In order to illustrate the use of abstraction when optimizing a program, consider the following rule defining the average of the elements of a given list (R_{15}) : $\texttt{average(L)} \rightarrow \texttt{sum(L)/len(L)}$. Observe that this definition traverses the same list twice, which can be avoided by applying tupling as follows:

1. Definition introduction: (R_{16}) : $\texttt{new}_2\texttt{(L)} \rightarrow \langle \texttt{sum(L)}, \texttt{len(L)} \rangle$
2. Unfold rule R_{16}:
   (R_{17}) : $\texttt{new}_2\texttt{([])} \rightarrow \langle \texttt{0}, \texttt{len([])} \rangle$ (R_{18}) : $\texttt{new}_2\texttt{([H|T])} \rightarrow \langle \texttt{H} + \texttt{sum(T)}, \texttt{len([H|T])} \rangle$
3. Normalize rules R_{17} and R_{18}:
   (R_{19}) : $\texttt{new}_2\texttt{([])} \rightarrow \langle \texttt{0}, \texttt{0} \rangle$ (R_{20}) : $\texttt{new}_2\texttt{([H|T])} \rightarrow \langle \texttt{H} + \texttt{sum(T)}, \texttt{s(len(T))} \rangle$
4. Abstract R_{20}:
   (R_{21}) $\texttt{new}_2\texttt{([H|T])} \rightarrow \langle \texttt{H} + \texttt{U}, \texttt{s(V)} \rangle$ where $\langle \texttt{U}, \texttt{V} \rangle = \langle \texttt{sum(T)}, \texttt{len(T)} \rangle$

[2] For a sequence of positions $P = \overline{p_n}$, we let $t[\overline{s_n}]_P = (((t[s_1]_{p_1})[s_2]_{p_2}) \ldots [s_n]_{p_n})$. By abuse, we denote $t[\overline{s_n}]_P$ by $t[s]_P$ when $s_1 = \ldots = s_n = s$, as well as $((t[s_1]_{P_1}) \ldots [s_n]_{P_n})$ by $t[\overline{s_n}]_{\overline{P_n}}$.

5. Fold R_{21} with R_{16}: (R_{22}) $\mathtt{new_2([H|T])} \rightarrow \langle \mathtt{H + U, s(V)} \rangle$ $\mathtt{where}$ $\langle \mathtt{U, V} \rangle = \mathtt{new_2(T)}$
6. Finally, after similar abstraction and folding steps on R_{15}, we obtain the desired definition (R_{23}) : $\mathtt{average(L)} \rightarrow \mathtt{U/V}$ $\mathtt{where}$ $\langle \mathtt{U, V} \rangle = \mathtt{new_2(L)}$, that only traverses a list once thanks to the use of the improved definition for $\mathtt{new_2}$ (rules R_{19} and R_{22}).

3 Structure of Transformation Strategies

In this section we decompose the internal structure of the transformation strategies of composition and tupling. Following [1] and [8], we focus our attention separately in the three transformation stages (definition introduction, unfolding and folding) shown in Table 1, where each phase consists of several steps done with the transformation rules described before.

Table 1. The `Tupling_Algorithm`

INPUT:	Initial Program $\mathcal{R}$ and Program Rule $R = (l \rightarrow r) \in \mathcal{R}$
BODY:	********** *DEFINITION INTRODUCTION PHASE* ********** 1. Let $T = \langle t_1, \ldots, t_n \rangle$ be the set (without repetitions) of pattern subterms sharing common variables in r 2. Apply the DEFINITION INTRODUCTION RULE to generate: $R_{def} = (f_{new}(\overline{x}) \rightarrow T)$ ******************* *UNFOLDING PHASE* ****************** 3. Let $\mathcal{R}_{unf} = \{R_{def}\}$ be a program 4. Repeat $\mathcal{R}_{unf}$=NORMALIZE(UNFOLD($\mathcal{R}_{unf}, \mathcal{R}$),$\mathcal{R}$) until every rule $R' \in \mathcal{R}_{unf}$ verifies TEST(R', R_{def})>0 ********************* *FOLDING PHASE* ****************** 5. Let $\mathcal{R}_{fold} = (\mathcal{R}_{unf} \cup R)$ be a program 6. For every rule $R' \in \mathcal{R}_{fold}$ verifying TEST(R, R_{def})=2 $\mathcal{R}_{fold} = (\mathcal{R}_{fold} - \{R'\}) \cup$ {FOLD(ABSTRACT(R', R_{def}))}
OUTPUT:	Transformed Program $\mathcal{R}_{fold}$

Definition Introduction Phase. The eureka generation phase, which is the key point for a transformation strategy to proceed [5,9,10,2]. For the case of the composition strategy, eureka definitions can be easily identified since they correspond to nested calls. However, for the tupling strategy the problem is much more difficult, mainly due to the fact that the (non-nested) calls to be tupled may be arbitrarily distributed into the rhs of a rule. Sophisticated static analyses techniques have been developed in the literature using dependency graphs ([6]), m-dags ([4]) and other intrincate structures. The main problems appearing in such approaches are that the analyses are not as general as wanted (they can fail even though the program admits tupling optimizations), and they are time and space consuming. In order to avoid these risks, our approach generates eureka definitions following a very simple strategy that obtains high levels of efficiency (see steps 1 and 2 in Table 1 and step 1 in Example 2). Since it is not usual that

Table 2. Function TEST

INPUT:	Original eureka	$f_{new}(\overline{x}) \rightarrow T$ such that $T = \langle t_1, \ldots, t_n \rangle$		
	Unfolded Rule	$\sigma(f_{new}(\overline{x})) \rightarrow T'$ such that $T' = \langle t'_1, \ldots, t'_n \rangle$		
BODY:	1. Let $S = \langle s_1, \ldots, s_m \rangle$ be the set (without repetitions) of pattern subterms sharing common variables in T'			
	2. Look if one of the following Stopping Conditions holds:			
	SC1:	$m < n$	$\Rightarrow$ return 1	* *base case definition* *
	SC2:	$m = n$ **and** $\theta(T) = S$	$\Rightarrow$ return 2	***** *foldability* *****
	SC3:	$m > n$	$\Rightarrow$ return 3	*** *new tupling loop* **
		Otherwise:	$\Rightarrow$ return 0	** *continue unfolding* *

terms to be tupled contain new function calls as parameters, we cope with this fact in our definition by requiring that only pattern subterms (sharing at least a common variable) of r be collected. On the contrary, the considered subterms would contain nested calls which should be more appropriately transformed by composition instead of tupling, as we will see in Section 4.

Unfolding Phase. During this phase, the eureka definition R_{def} generated in the previous phase, is unfolded possibly several times (at least once) using the original definitions of program $\mathcal{R}$, and returning a new program $\mathcal{R}_{unf}$ which represents the unfolded definition of f_{new}. In each iteration of this phase, once a rule in $\mathcal{R}_{unf}$ is unfolded, it is removed and replaced with the rules obtained after unfolding it in the resulting program $\mathcal{R}_{unf}$, which is dynamically updated in our algorithm. The key point to stop the unfolding loop (see steps 3 and 4 in table 1, where unfolded rules are normalized as much as possible once obtained) is the use of function TEST described in table 2, which compares the tuples of patterns sharing variables of eurekas and unfolded rules in order to decide when a base case as been reached, a regularity has been found or a new tupling cycle must be (incrementally) re-initiated.

Folding Phase. The aim of this phase (steps 5 and 6 of table 1) is not only to obtain efficient recursive definitions of the new symbol f_{new} (initially defined by the eureka $\mathcal{R}_{def}$), but also to redefine old function symbols in terms of the optimized definition of f_{new}. This fact depends on the rule R to be folded, which may belong to the unfolded program obtained in the previous phase ($\mathcal{R}_{unf}$), or to the original program ($\mathcal{R}$), respectively. The application of a folding step on a given rule is similar to rename a subterm in its rhs with an instance of $f_{new}(\overline{x})$. For the case of the composition strategy, folded (renamed) subterms are obviously nested (composed) expressions. However, before performing folding steps when applying tupling, we first must to generate the tuple to be folded by means of abstraction steps as follows. Firstly we consider the case $R \in \mathcal{R}_{unf}$. Remember that $R_{def} = (f_{new}(\overline{x}) \rightarrow T)$ where $T = \langle t_1, \ldots, t_n \rangle$. If $R = (\sigma(f_{new}(\overline{x})) \rightarrow r')$ satisfies the foldability condition (SC2) explained in the previous phase then, it is possible to abstract R accordingly to tuple T and generate the new rule $\sigma(f_{new}(\overline{x})) \rightarrow r'[\overline{z_j}]_{\overline{P_j}}$ *where* $\langle z_1, \ldots, z_n \rangle = \theta(\langle t_1, \ldots, t_n \rangle)$, as illustrates step

4 and R_9 in Example 2. After a subsequent folding step using R_{def} (see rule R_{22} and step 5 in Example 2), we obtain a recursive definition of f_{new}, as desired. The case when $R \in \mathcal{R}$ is perfectly analogous (see rule R_{23} in the same example).

4 Incremental Tupling with Simplification Pre-process

We propose now three final improvements which can be naturally embedded in our tupling algorithm as shown in table 3. The global algorithm submits all rules of a given program to the transformation process (step 2 in the table). Before being treated by tupling (step 2.3), each program rule R, is firstly simplified by normalization (generating rule R' in step 2.1) and composition (generating rule R'' in step 2.2). Finally, step 2.4 introduces our notion of incremental tupling.

Improvement Based on "Normalization+Tupling". As described in Section 3, normalization is a powerful tool systematically used for simplifying rules during the unfolding phase. Moreover, its benefits also hold when it is applied before starting a (composition and) tupling loop, as the following example shows.

Example 3. The following program defines function `fact` for computing the factorial of a given number, and `fli` and $\mathtt{fli^{ev}}$ for generating lists with factorials of consecutive natural numbers (only even numbers are considered in $\mathtt{fli^{ev}}$):

$$\begin{array}{llll}
(R_{36}): & \mathtt{fact(0)} \to \mathtt{s(0)} & (R_{37}): & \mathtt{fact(s(N))} \to \mathtt{s(N)} * \mathtt{fact(N)} \\
(R_{38}): & \mathtt{fli(0)} \to [\,] & (R_{39}): & \mathtt{fli(s(N))} \to [\mathtt{fact(s(N))}|\mathtt{fli(N)}] \\
(R_{40}): & \mathtt{fli^{ev}(0)} \to [\,] & (R_{41}): & \mathtt{fli^{ev}(s(s(X)))} \to [\mathtt{fact(s(s(X)))}|\mathtt{fli^{ev}(X)}] \\
(R_{42}): & \mathtt{fli^{ev}(s(0))} \to [\,] & &
\end{array}$$

If we try to improve both definitions for `fli` and $\mathtt{fli^{ev}}$, our tupling algorithm would proceed by creating eurekas with tuples $\langle \mathtt{fact(s(X))}, \mathtt{fli(X)} \rangle$ and $\langle \mathtt{fact(s(s(X)))}, \mathtt{fli^{ev}(X)} \rangle$, respectively, and continuing with never ending unfolding loops, as the reader may easily check. In particular, the second case has been proposed in [6] as a non trivial example to be solved by tupling. However, in contrast with the rather complicated solution (based on involved eureka analyses) presented there, we propose a much simpler way to solve the

Table 3. Improved Algorithm: Normalization+Composition+Incremental Tupling

INPUT:	Original Program $\mathcal{R}$	
BODY:	1.	Initialize program $\mathcal{R}' = \mathcal{R}$
	2.	For each rule $R \in \mathcal{R}'$ do
	2.1.	$\{R'\}$=NORMALIZE($\{R\}, \mathcal{R}'$)
	2.2.	$\mathcal{R}'$=Composition_Algorithm($(\mathcal{R}' - \{R\}) \cup \{R'\}, R'$)
	2.3.	$\mathcal{R}'$=Tupling_Algorithm($\mathcal{R}', R''$)
	2.4.	For every rule $R^* \in \mathcal{R}'$ verifying TEST(R^*, R_{def})=3
		$\mathcal{R}'$=Tupling_Algorithm($\mathcal{R}', R^*$)
OUTPUT:	Transformed Program $\mathcal{R}'$	

problem. The idea is to simply normalize rules R_{39} and R_{41} before being transformed by tupling. So, after normalizing R_{39} we obtain the new rule (R_{43}) : $\mathtt{fli(s(X))} \rightarrow \mathtt{[s(X)*fact(X)|fli(X)]}$. Moreover a normalizing step on R_{41} generates (R_{44}) : $\mathtt{fli^{ev}(s(s(X)))} \rightarrow \mathtt{[s(s(X))*fact(s(X))|fli^{ev}(X)]}$, which also admits a final normalizing step to become (R_{45}) : $\mathtt{fli^{ev}(s(s(X)))} \rightarrow \mathtt{[s(s(X))*s(X)*fact(X)|fli^{ev}(X)]}$. Now, starting with this pair of fully normalized rules, we can generate the respective pair of appropriate eureka definitions (R_{46}) : $\mathtt{new_5(X)} \rightarrow \langle\mathtt{fact(X),fli(X)}\rangle$ and (R_{47}) : $\mathtt{new_5^{ev}(X)} \rightarrow \langle\mathtt{fact(X),fli^{ev}(X)}\rangle$, which can be now easily treated by tupling.

Improvement Based on "Composition+Tupling". On the other hand, we consider now a second class of programs where the optimization to be pursued could not proceed by considering only patterns subterms (i.e., not containing nested calls) to be tupled, since, for those cases, the tuple generated by our method only would contain an unique element. Remember that the composition strategy optimizes the definition of nested expressions by generating a single new function definition. Hence, after applying the composition strategy as much as possible, new patterns (representing calls to enhanced eureka definition) may arise on transformed rules, and this is just we need before initiating tupling.

Example 4. Continuing with the examples of Section 2, consider the following definition (R_{48}) : $\mathtt{f(N)} \rightarrow \mathtt{sum(gen(N))/len(gen(N))}$. Observe that this function computes the formula $\mathtt{f(n)} = (\sum_{\mathtt{i=0}}^{\mathtt{n}} \mathtt{i})/\mathtt{n}$ in a very inneficient way: a same list is created (two calls to gen) and traversed (one call to sum and one more to len) twice!. Unfortunately, our tupling strategy is not able to optimize the previous definition, since it only considers the unique subterm gen(N) when building the initial eureka definition. However, we observe that the composition strategy can be applied twice to expressions sum(gen(N)) and len(gen(N)) on rule R_{48}. For the first case we obtain the optimized definition of $\mathtt{new_1}$ seen in Example 1 and for the second one it is easy to build (by composition) the following new definition: (R_{49}) : $\mathtt{new_6(0)} \rightarrow \mathtt{s(0)}$, (R_{50}) : $\mathtt{new_6(s(N))} \rightarrow \mathtt{s(new_6(N))}$. After this pre-process, the original definition for f becomes $\mathtt{f(N)} \rightarrow \mathtt{new_1(N)/new_6(N)}$ which allows us to start a tupling loop in order to finally obtain the following (fully optimized, without the need of creating and traversing intermediate lists) definition: $\mathtt{f(N)} \rightarrow \mathtt{U/V}$ where $\langle\mathtt{U,V}\rangle = \mathtt{new_7(N)}$, having that $\mathtt{new_7(0)} \rightarrow \langle\mathtt{0,0}\rangle$ and $\mathtt{new_7(s(N))} \rightarrow \langle\mathtt{s(N)+U,s(V)}\rangle$ where $\langle\mathtt{U,V}\rangle = \mathtt{new_7(N)}$.

Improvement Based on "Tupling+Tupling". Our incremental version of the tupling algorithm highly increases its power, without seriously altering its core, by simply generating, optimizing and chaining several eureka definitions.

Example 5. The classical Hanoi's Towers problem can be solved in a natural but inefficient way by using the pair of rules (R_{51}) : $\mathtt{h(0,A,B,C)} \rightarrow \mathtt{[]}$ and (R_{52}) : $\mathtt{h(s(N),A,B,C)} \rightarrow \mathtt{app(h(N,A,C,B),[mov(A,B)|h(N,C,B,A)])}$ (together with rules defining the typical app function for concatenating lists). This program, with exponential complexity due to the two recursive calls in the rhs of rule R_{52}, can be optimized by tupling as follows:

1. Definition Introduction (R_{53}) : $\mathtt{new_8(N,A,B,C)} \rightarrow \langle\mathtt{h(N,A,C,B),h(N,C,B,A)}\rangle$
2. During the unfolding phase, we generate the pair of rules:

(R_{54}) : $\mathtt{new_8(0,A,B,C)} \rightarrow \langle[\,],[\,]\rangle$ **SC1!**
(R_{55}) : $\mathtt{new_8(s(N),A,B,C)} \rightarrow \langle\mathtt{app(h(N,A,B,C),[mov(A,C)|h(N,B,C,A)])},$
$\mathtt{app(h(N,C,A,B),[mov(C,B)|h(N,A,B,C)])}\rangle$ **SC3!**

At this point R_{54} represents a case base definition for $\mathtt{new_8}$. Regarding R_{55}, we observe that its rhs contains four calls (sharing common variables) to h, where only three of them are differents. So, if we force a premature application of abstraction+folding steps, the recursive definition of $\mathtt{new_8}$ would have two calls: one for $\mathtt{new_8}$ and one for h, which does not represent an enhanced definition for $\mathtt{new_8}$. On the other hand, it would be more appropriate to re-generate a new tupling process in order to optimize R_{55}, since now our eureka generator is able to produce an eureka with three calls (patterns sharing common variables), as desired. Hence, our incremental algorithm simply consists in generating a new tupling cycle when the stopping condition **SC3** described in Table 2 is reached. In our example, the new tupling process will consider as initial eureka the new rule (R_{56}) : $\mathtt{new_9(N,A,B,C)} \rightarrow \langle\mathtt{h(N,A,B,C),h(N,B,C,A),h(N,C,A,B)}\rangle$. Moreover, the unfolding phase will generate new pair of rules:

(R_{57}) : $\mathtt{new_9(0,A,B,C)} \rightarrow \langle[\,],[\,],[\,]\rangle$ **SC1!**
(R_{58}) : $\mathtt{new_9(s(N),A,B,C)} \rightarrow \langle\mathtt{app(h(N,A,C,B),[mov(A,B)|h(N,C,B,A)])},$
$\mathtt{app(h(N,B,A,C),[mov(B,C)|h(N,A,C,B)])},$
$\mathtt{app(h(N,C,B,A),[mov(C,A)|h(N,B,A,C)])}\rangle$ **SC2!**

At this moment, the unfolding process must finish since R_{57} represents a base case definition for $\mathtt{new_9}$ and R_{58} satisfies the stopping condition (foldability) **SC2** in Table 2). Now, we proceed with the folding phase obtaining:

(R_{59}) : $\mathtt{new_9(s(N),A,B,C)} \rightarrow \langle\mathtt{app(Z_1,[mov(A,B)|Z_2]),\ app(Z_3,[mov(B,C)|Z_1])},$
$\mathtt{app(Z_2,[mov(C,A)|Z_3])}\rangle$
$\mathtt{where}\ \langle\mathtt{Z_1,Z_2,Z_3}\rangle = \mathtt{new_9(N,A,C,B)}$
(R_{60}) : $\mathtt{new_8(s(N),A,B,C)} \rightarrow \langle\mathtt{app(Z_1,[mov(A,C)|Z_2]),app(Z_3,[mov(C,B)|Z_1])}\rangle$
$\mathtt{where}\ \langle\mathtt{Z_1,Z_2,Z_3}\rangle = \mathtt{new_9(N,A,B,C)}$

Thanks to this re-application of the whole tupling algorithm, we have got improved definitions for both eurekas. Then, we can recover the first tupling loop at the point we left it, in order to reuse the enhanced definition of $\mathtt{new_8}$ when re-defining h (by appropriately abstracting and folding R_{52}) as: $\mathtt{h(s(N),A,B,C)} \rightarrow \mathtt{app(Z_1,mov(A,B) : Z_2)}$ where $\langle\mathtt{Z_1,Z_2}\rangle = \mathtt{new_8(N,A,B,C)}$. Now, this last rule, together with $R_{51}, R_{54}, R_{57}, R_{59}$ and R_{60}, represents an improvement w.r.t. the original program thanks to the chained use of enhanced definitions for the new symbols $\mathtt{new_8}$ and $\mathtt{new_9}$.

To finish this section, we remark that the massive use of function TEST together with the extensive exploration of patterns sharing common variables, are the more significant and distinctive points of our improved approach to automatic tupling. Similarly to other heuristics shown in the literature, we assume that the algorithm aborts when it overflows a maximal number of allowed unfolding and tupling iterations (i.e., when no enough stopping conditions **SC1** -base cases- and **SC2** -foldable regularities- have been found).

5 Conclusions

Tupling is a powerful optimization strategy which can be achieved by fold/unfold transformations and produces better gains in efficiency than other simpler ones such as composition. As it is well-known in the literature, tupling is very complicated and many automatic tupling algorithms either result in high runtime cost or they succeed only for a restricted class of programs. Starting with fully automatic composition and tupling algorithms that we have previously designed in recent works, our approach refines them and removes some of these limitations.

It is important to remark that the idea of considering only patterns sharing common variables has very nice properties: it is a very easy, purely syntactic method, that can be efficiently repeated along the three transformation phases, i.e., not only for generating eureka definitions but also during the unfolding phase while searching for regularities and finally, for the application of abstraction and folding steps. Moreover, it is also useful for testing when a new tupling cycle must be initiated, hence producing chained definitions of eureka symbols.

We have also shown how to increase both the speed-up of the process and the class of programs to be optimized by simply performing some simplification pre-processes on the original program before applying tupling. In particular, apart that normalization is useful during the unfolding phase, its use also generates programs more amenable to higher optimizations. Similarly, composition is not only interesting for producing important optimizations on a given program, but also for giving more chances to the tupling strategy to successfully proceed.

References

1. M. Alpuente, M. Falaschi, G. Moreno, and G. Vidal. An Automatic Composition Algorithm for Functional Logic Programs. In *Proc. of the 27th Conf. on Current Trends in Informatics, SOFSEM'2000*, pages 289–297. Springer LNCS 1963, 2000.
2. M. Alpuente, M. Falaschi, G. Moreno, G. Vidal. Rules+Strategies for Transforming Lazy Functional Logic Programs. *Theoretical Computer Science,* 311:479–525,2004.
3. S. Antoy, R. Echahed, and M. Hanus. A Needed Narrowing Strategy. In *Proc. 21st ACM Symp. on POPL, Portland,* pages 268–279, New York, 1994. ACM Press.
4. R.S. Bird. Tabulation techniques for recursive programs. *ACM Computing Surveys,* 12(4):403–418, 1980.
5. R.M. Burstall and J. Darlington. A Transformation System for Developing Recursive Programs. *Journal of the ACM*, 24(1):44–67, 1977.
6. W. Chin. Towards an Automated Tupling Strategy. In *Proc. of Partial Evaluation and Semantics-Based Program Manipulation, 1993,* pages 119–132. ACM, 1993.
7. S. González and G. Moreno. Incremental Tupling with Simplification Pre-Process. Technical Report DIAB-04-05-1, UCLM, 2004.

8. G. Moreno. Automatic Optimization of Multi-Paradigm Declarative Programs. In *Proc. of the 8th Ibero–American Conference on Artificial Intelligence, IBERAMIA'2002*, pages 131–140. Springer LNAI 2527, 2002.
9. A. Pettorossi and M. Proietti. Rules and Strategies for Transforming Functional and Logic Programs. *ACM Computing Surveys,* 28(2):360–414, 1996.
10. H. Tamaki and T. Sato. Unfold/Fold Transformations of Logic Programs. In *Proc. of* 2^{nd} *Int'l Conf. on Logic Programming, Uppsala, Sweden,* pages 127–139, 1984.

Polynomial Classes of Boolean Formulas for Computing the Degree of Belief

Guillermo De Ita Luna*

Institute in Astrophysics, Optics and Electronics
deita@ccc.inaoep.mx

Abstract. Given a knowledge base Σ and a formula F both in propositional Conjunctive Form, we address the problem of designing efficient procedures to compute the degree of belief in F with respect to Σ as the conditional probability $P_{F|\Sigma}$. Applying a general approach based on the probabilistic logic for computing the degree of belief $P_{F|\Sigma}$, we can determine classes of conjunctive formulas for Σ and F in which $P_{F|\Sigma}$ can be computed efficiently. It is known that the complexity of computing $P_{F|\Sigma}$ is polynomially related to the complexity of solving the #SAT problem for the formula $\Sigma \wedge F$. Therefore, some of the above classes in which $P_{F|\Sigma}$ is computed efficiently establish new polynomial classes given by $\Sigma \cup F$ for the #SAT problem and, consequently, for many other related counting problems.

Keywords: #SAT Problem, Degree of Belief, Updating Beliefs, Approximate Reasoning.

1 Introduction

The general approach used here to compute the degree of belief, consists of assigning an equal degree of belief to all basic "situations". In this manner, we can compute the probability that Σ (an initial knowledge base which involves n variables) will be satisfied, denoted by P_Σ, as: $P_\Sigma = Prob(\Sigma \equiv \top) = \frac{\mu(\Sigma)}{2^n}$, where $\top$ stands for the Truth value, *Prob* is used to denote the probability, and $\mu(\Sigma)$ denotes the number of models that Σ has.

We are interested in the computational complexity of computing the degree of belief in a propositional formula F with respect to Σ, such as the fraction of models of Σ that are consistent with the query F, that is, the conditional probability of F with respect to Σ, denoted by $P_{F|\Sigma}$, and computed as: $P_{F|\Sigma} = Prob((\Sigma \wedge F) \equiv \top \mid \Sigma \equiv \top) = \frac{\mu(\Sigma \wedge F)}{\mu(\Sigma)}$.

Conditional probabilities are key for reasoning because they formalize the process of accumulating evidence and updating probabilities based on new evidence. We will use the probability theory as a formal means of manipulating the degrees of belief. The inference of a degree of belief is a generalization of

* This research is partly supported by the Autonomous University of Puebla - BUAP.

deductive inference which can be used when the knowledge base is augmented by, e.g., statistical information, or in an effort to avoid the computationally hard task of deductive inference [8].

As it is known, the complexity of computing the degree of belief, $P_{F|\Sigma}$, is polynomially related to the complexity of counting the number of models of $\Sigma \wedge F$, which directs us to the #SAT problem.

The #SAT problem consists of counting the number of satisfying assignments for a given propositional formula. #SAT is at least as hard as the decision problem SAT, but in many cases, even when SAT is solved in polynomial time, no computationally efficient method is known for counting the number of distinct solutions.

For example, the 2-SAT problem, SAT restricted to consider a conjunction of 2-clauses (binary clauses), it can be solved in linear time. However, the corresponding "counting" problem #2-SAT is a #P-complete problem. This also applies to the restrictions of #SAT for monotone 2-clauses, 2-HORN, and $(2, 3\mu)$-CF (conjunction of 2-clauses where each variable appears three times at most), which are all #P-complete problems [8, 9].

The maximal class of Conjunctive forms where #SAT can be solved in polynomial time is for the class $(2, 2\mu)$-CF (conjunction of binary clauses where each variable appears two times at most) [8, 9]. In order to efficiently compute the degree of belief in F, given an initial knowledge base Σ, we would then consider Σ as a $(2, 2\mu)$-CF. In this sense, we will develop an efficient procedure in chapter 3 for solving #SAT for formulas in $(2, 2\mu)$-CF. Furthermore, we will analyze how far we can extend the computing of the degree of belief $P_{F|\Sigma}$ into the class of efficient procedures.

The research presented here follows the line pointed out by Eiter and Gottlob [3], Papadimitriou [5], Darwiche [2], Zanuttini [10] and many others who have analyzed problems arising from deductive inference, such as searching for explanations, approximate reasoning and computing the degree of belief. These works try to differentiate the classes of Boolean formulas where such problems can be solved efficiently from those classes where such problems present an exponential complexity.

2 Notation and Preliminaries

Let $X = \{x_1, \ldots, x_n\}$ be a set of n *Boolean variables*. A *literal* is either a variable x or a negated variable $\overline{x}$. The pair $\{x, \overline{x}\}$ is *complementary*. We denote $\overline{l}$ as the negation of the literal l. We use $v(l)$ to indicate the variable involved by the literal l. As it is usual, for each $x \in X$, $x^0 = \overline{x}$ and $x^1 = x$. We use $\perp$ and $\top$ as two constants in the language which represent the truth values *false* and *true*, respectively.

A *clause* is a disjunction of literals. For $k \in I\!N$, a ***k*-clause** is a clause consisting of exactly k literals and, **$(\leq k)$-clause** is a clause with k literals at the most. A unitary clause has just one literal and a binary clause has exactly two literals.

A variable $x \in X$ *appears* in a clause c if either x or $\overline{x}$ is an element of c. Let $v(c) = \{x \in X | x \text{ appears in } c\}$.

A *conjunctive form* (CF) is a conjunction of clauses (we will also consider a CF as a set of clauses). A k-CF is a CF containing only k-clauses and, $(\leq k)$-CF denotes a CF containing clauses with at most k literals. A $k\mu$-CF is a formula in which no variable occurs more than k times. A $(k, s\mu)$-CF is a k-CF such that each variable appears no more than s times, similarly a $(\leq k, s\mu)$-CF is a $(\leq k)$-CF where each variable appears s times at most. In this sense we have a hierarchy given by the number of ocurrences by variable, where $(k, s\mu)$-CF is a restriction of $(k, (s+1)\mu)$-CF, and a hierarchy given by the number of literals by clause, where $(\leq k, s\mu)$-CF is a restriction of $(\leq (k+1), s\mu)$-CF.

For any CF F, let $v(F) = \{x \in X | x \text{ appears in } F\}$. On the other hand, a CF F is of the type $\mathcal{F}(m, n)$ if F consists of m clauses which involves n variables.

An assignment s for F is a function $s : v(F) \to \{0, 1\}$. An *assignment* also can be considered as a set of literals where there are no complementary pairs of literals. If l is an element of an assignment, then the assignment makes l *true* and makes $\overline{l}$ *false*. A clause c is *satisfied* by the assignment s if and only if $c \cap s \neq \emptyset$, otherwise we can say that c is *contradicted* or *falsified* by s.

A CF F is *satisfied* by an assignment s if each clause in F is satisfied by s. F is *contradicted* by s if any clause in F is contradicted by s. A model of F is an assigment over $v(F)$ that satisfies F. Let $M(F)$ be the set of models that F has over $v(F)$. F is a *contradiction* or *unsatisfiable* if $M(F) = \emptyset$. Let $\mu_{v(F)}(\Sigma) = | M(\Sigma) |$ be its cardinality. Let $K_{v(F)}(F)$ be the set of assignments over $v(F)$ which unsatisfies F. When $v(F)$ will clear from the context, we will explicitly omit it as a subscript.

If $F_1 \subset F$ is a formula consisting of some clauses of F, then $v(F_1) \subset v(F)$. An assignment over $v(F_1)$ is a *partial* assignment over $v(F)$. Indeed, any assignment over $v(F_1)$ has $2^{|v(F)| - |v(F_1)|}$ extensions as assignments over $v(F)$.

Let #*LANG*-SAT be the notation for the #SAT problem for propositional formulas in the class *LANG*-CF, i.e. #2-SAT denotes #SAT for formulas in 2-CF, while $\#(2, 2\mu)$-SAT denotes #SAT for formulas in the class $(2, 2\mu)$-CF.

3 Counting the Number of Models of a Formula

Let Σ be a CF. If $\mathcal{F} = \{F_1, \ldots, F_r\}$ is a partition of Σ (over the set of clauses appearing in Σ), i.e. $\bigcup_{\rho=1}^{r} F_\rho = \Sigma$ and $\forall \rho_1, \rho_2 \in [\![1, r]\!], [\rho_1 \neq \rho_2 \Rightarrow F_{\rho_1} \cap F_{\rho_2} = \emptyset]$, we will say that $\mathcal{F}$ is a *partition in connected components* of Σ if $\mathcal{V} = \{v(F_1), \ldots, v(F_r)\}$ is a partition of $v(\Sigma)$.

If $\{F_1, \ldots, F_r\}$ is a partition in connected components of Σ, then:

$$\mu_{v(\Sigma)}(\Sigma) = \left[\mu_{v(F_1)}(F_1)\right] \cdot \ldots \cdot \left[\mu_{v(F_r)}(F_r)\right] \tag{1}$$

In order to compute $\mu(\Sigma)$, we should first determine the set of connected components of Σ, and this procedure can be done in linear time [1,9]. If Σ is a 2-CF, it is easy to find the connected components of Σ, for this, let the undirected *clause-graph* G_Σ given by: each clause of Σ is a node in G_Σ and there is an edge: (c, c') labeled by x if x is the common variable between the clauses c and c', $v(c) \cap v(c') = \{x\}$. The different connected components of G_Σ conform the partition of Σ in its connected components. From now on, when we mention a formula Σ, we can suppose that all of its clauses are connected (there is a path connecting whatever two clauses of Σ).

3.1 An Algorithm for #(2, 2μ)-SAT

The problem of calculating #(2, 2μ)-SAT is reduced to designing procedures for computing the number of models of each connected component type. We will suppose that G_Σ is the *clause-graph* of a connected component type given by Σ, and we will present the different cases for computing $\mu(\Sigma)$.

i) If G_Σ consists of just one node, its associated subformula Σ consists of just one binary clause and $\mu(\Sigma) = 3$.
ii) If $\Sigma = \{c, c'\}$ where $v(c) = v(c')$, then G_Σ is a cycle with two nodes and $\mu(\Sigma) = 2$, since only two assignments over $v(c)$ falsify both c and c'.
iii) Let $G_\Sigma = (V, E)$ be a linear chain, where $| V | = | \{c_1, ..., c_m\} | = m$, where $| v(c_i) \cap v(c_{i+1}) | = 1$, $i = 1, ..., (m-1)$. Let us write Σ, without a loss of generality (ordering the clauses and its literals, if it were necessary), as: $\Sigma = \left\{\{y_0^{\epsilon_1}, y_1^{\delta_1}\}, \{y_1^{\epsilon_2}, y_2^{\delta_2}\}, \ldots, \{y_{m-1}^{\epsilon_m}, y_m^{\delta_m}\}\right\}$, where $\delta_i, \epsilon_i \in \{0, 1\}$, $i = 1, ..., m$.
We will compute $\mu(\Sigma)$ by calculating the value $\mu(f_i)$ in each step, where f_i is a family of clauses of Σ built as follows: $f_i = \{c_j\}_{j \le i}$, $i = 1, ..., m$. Then $f_i \subset f_{i+1}$, $i = 1, ..., m-1$. Let $M(f_i) = \{\text{assignments over } v(f_i) \text{ satisfying } f_i\}$, $A_i = \{s \in M(f_i) | y_i \in s\}$, $B_i = \{s \in M(f_i) | \overline{y_i} \in s\}$. And let $\alpha_i = |A_i|$; $\beta_i = |B_i|$, $\mu_i = |M(f_i)| = \alpha_i + \beta_i$.
We will calculate the pair (α_i, β_i) in each step $i \in [\![1, m]\!]$ according to the signs (ϵ_i, δ_i) of the literals in the clause c_i, as follows: For the first clause the initial pair (α_1, β_1), is: $(\alpha_1, \beta_1) = \begin{cases} (1,2) & \text{if } \delta_1 = 0, \\ (2,1) & \text{if } \delta_1 = 1. \end{cases}$.
In a recurrent way, for $i \in [\![2, m]\!]$, we can determine the values (α_i, β_i), as:

$$\begin{array}{ll} (\ \beta_{i-1}, \alpha_{i-1} + \beta_{i-1}\ ) & \text{if } (\epsilon_i, \delta_i) = (0,0) \\ (\ \alpha_{i-1} + \beta_{i-1}, \beta_{i-1}\ ) & \text{if } (\epsilon_i, \delta_i) = (0,1) \\ (\ \alpha_{i-1}, \alpha_{i-1} + \beta_{i-1}\ ) & \text{if } (\epsilon_i, \delta_i) = (1,0) \\ (\ \alpha_{i-1} + \beta_{i-1}, \alpha_{i-1}\ ) & \text{if } (\epsilon_i, \delta_i) = (1,1) \end{array} \tag{2}$$

As $\Sigma = f_m$ and $|M(f_i)| = \mu_i = \alpha_i + \beta_i$ then $\mu(\Sigma) = |M(f_m)| = \mu_m = \alpha_m + \beta_m$.

Example 1. *Suppose that F is a* monotone *2-CF and G_F is a linear chain. I.e* $F = \{(x_0, x_1), (x_1, x_2), \ldots, (x_{m-1}, x_m)\}$. $(\alpha_1, \beta_1) = (2, 1)$ *since* $\delta_1 = 1$.

$$\begin{array}{llll} \mu_1 & = \alpha_1 + \beta_1 = 2 + 1 & = 3, & \\ \mu_2 & = \alpha_2 + \beta_2 = (\alpha_1 + \beta_1) + \alpha_1 & = \mu_1 + \alpha_1 & = 3 + 2 = 5, \\ \forall i \ge 2 : \mu_i = \alpha_i + \beta_i & = (\alpha_{i-1} + \beta_{i-1}) + \alpha_{i-1} & = \mu_{i-1} + \alpha_{i-1} & = \mu_{i=1} + \mu_{i-2}. \end{array}$$

The Fibonacci series appears!. I.e. Applying the Fibonacci series until $m = 5$, *we obtain the values* $(\alpha_i, \beta_i), i = 1, ..., 5$: $(2,1) \rightarrow (3,2) \rightarrow (5,3) \rightarrow (8,5) \rightarrow (13,8)$ *and* $\mu(F) = \mu_5 = 21$.

iv) Let $G_\Sigma = (V, E)$ be a cycle of m nodes, then all the variables in $v(\Sigma)$ appear two times, and $| V | = m = n = | E |$. Ordering the clauses in Σ in such a way that $| v(c_i) \cap v(c_{i+1}) | = 1$, and $c_{i_1} = c_{i_2}$ whenever $i_1 \equiv i_2 \bmod m$, hence $y_0 = y_m$, then $\Sigma = \left\{ c_i = \{ y_{i-1}^{\epsilon_i}, y_i^{\delta_i} \} \right\}_{i=1}^{m}$, where $\delta_i, \epsilon_i \in \{0, 1\}$.

The first clause $c_1 = \{ y_m^{\epsilon_1}, y_1^{\delta_1} \}$ has three satisfying assignments given by $s_1 = \{ y_m^{1-\epsilon_1}, y_1^{\delta_1} \}$, $s_2 = \{ y_m^{\epsilon_1}, y_1^{1-\delta_1} \}$ and $s_3 = \{ y_m^{\epsilon_1}, y_1^{\delta_1} \}$. If $\delta_1 = \epsilon_2$ we define:

$$(\alpha_1, \beta_1)(s_1) = \begin{cases} (1,0) & \text{if } \delta_1 = 1, \\ (0,1) & \text{otherwise.} \end{cases}$$

$$(\alpha_1, \beta_1)(s_2) = \begin{cases} (0,1) & \text{if } \delta_1 = 1, \\ (1,0) & \text{otherwise.} \end{cases}$$

$$(\alpha_1, \beta_1)(s_3) = \begin{cases} (1,0) & \text{if } \delta_1 = 1, \\ (0,1) & \text{otherwise.} \end{cases}$$

For each $\lambda = 1, 2, 3$ calculate $(\alpha_i, \beta_i)(s_\lambda)$, with $i = 2, \ldots, m$, as it is indicated by the recurrence (2) and let $t(s_\lambda) = \begin{cases} \alpha_m & \text{if } y_m \in s_\lambda, \\ \beta_m & \text{if } \overline{y}_m \in s_\lambda. \end{cases}$
Then $\mu(\Sigma) = t(s_1) + t(s_2) + t(s_3)$.

For $i = 1, ..., m-1$ we will associate each pair (α_i, β_i) with the label y_i which is the common variable between the clauses c_i and c_{i+1}. Notice that if G_Σ is a linear chain then y_m is the label of (α_m, β_m) and y_0 is no the label of any pair, while if G_Σ is a cycle then all the variables in F appear as a label of any pair (α_i, β_i), including $y_0 = y_m$ associated to (α_m, β_m).

Example 2. *Let* $F = \{ (\overline{y}_0, y_1), (\overline{y}_1, y_2), (\overline{y}_2, y_3), (\overline{y}_3, y_4), (\overline{y}_4, y_0) \}$ *an* implicative 2-*CF. As* $c_1 = (\overline{y}_0, y_1)$ *then:* $s_1 = (y_0, y_1)$; $s_2 = (\overline{y}_0, \overline{y}_1)$; $s_3 = (\overline{y}_0, y_1)$, *and,*

$(\alpha_1, \beta_1)(s_1) = (1,0) \rightarrow (\alpha_2, \beta_2)(s_1) = (1,0) \rightarrow (\alpha_3, \beta_3)(s_1) = (1,0) \rightarrow (\alpha_4, \beta_4)(s_1) = (1,0) \rightarrow (\alpha_5, \beta_5)(s_1) = (1,0)$, *and* $t(s_1) = \alpha_5 = 1$ *since* $y_0 \in s_1$.

$(\alpha_1, \beta_1)(s_2) = (0,1) \rightarrow (\alpha_2, \beta_2)(s_1) = (1,1) \rightarrow (\alpha_3, \beta_3)(s_2) = (2,1) \rightarrow (\alpha_4, \beta_4)(s_2) = (3,1) \rightarrow (\alpha_5, \beta_5)(s_2) = (4,1)$, *then* $t(s_2) = \beta_5 = 1$ *since* $\overline{y}_0 \in s_2$.

And the series $(\alpha_i, \beta_i), i = 2, ..., 5$ *is the same both* (s_3) *as well as* (s_1), *then* $(\alpha_5, \beta_5)(s_3) = (1,0)$, *and* $t(s_3) = \beta_5 = 0$ *since* $\overline{y}_0 \in s_3$. *Finally,* $\mu(\Sigma) = t(s_1) + t(s_2) + t(s_3) = 1 + 1 + 0 = 2$.

As can be seen, the procedures for computing $\mu(\Sigma)$ being Σ a $(2, 2\mu)$-CF have polynomial time complexity since these procedures are based on applying (2) each time that each node of a connected component is visited. There are other procedures for computing $\mu(\Sigma)$ when Σ is a $(2, 2\mu)$-CF [8, 9], but these last proposals don't distinguish the models in which a variable x takes value 1 of those models in which the same variable x takes value 0, situation which is made explicit in our procedure through the pair (α, β) which is labeled by x. This distinction over the set of models of Σ is essential when we want to count the new set of models for Σ union a new formula F, how we will see in the next sections.

4 Efficient Computing of the Degree of Belief

Let Σ a $(2, 2\mu)$-CF, we are considering that Σ is consistent and $\mu(\Sigma) > 0$ and then $P_{F|\Sigma} = \frac{\mu(\Sigma \wedge F)}{\mu(\Sigma)}$ is well-defined. We will show here how to compute $\mu(\Sigma \wedge F)$; for this, we consider the different cases for F.

Let $F = \{(l)\}$, where $v(l) \in v(\Sigma)$. We can suppose that we had ordered the clauses in Σ, as: $\Sigma = \left\{c_i = \{y_{i-1}^{\epsilon_i}, y_i^{\delta_i}\}\right\}_{i=1}^{m}$ $\delta_i, \epsilon_i \in \{0,1\}, v(c_i) \cap v(c_{i+1}) = \{y_i\}$. In order to compute $\mu(\Sigma \wedge (l))$, while we are applying (2), we determine the index j, $1 \leq j \leq m$ such that the label associated to (α_j, β_j) is $v(l)$, that is, $v(l) = y_j$, and we modify (α_j, β_j), as: $(\alpha_j, \beta_j) = \begin{cases} (0, \beta_j) & \text{if } l \text{ appears negative in } F, \\ (\alpha_j, 0) & \text{otherwise.} \end{cases}$

Notice that if G_Σ is a cycle, the above modification is applied to each of the three pairs: $(\alpha_j, \beta_j)(s_k), k = 1, 2, 3.$ The case $(0, \beta_j)$ results because we are considering that the unitary clause $(\overline{v(l)})$ belongs to F and then $v(l)$ can not be set 'true' in any model of $\Sigma \wedge F$. Similarly, $(\alpha_j, 0)$ comes from considering that $(v(l))$ appears in F and then $v(l)$ can not be set 'false' in any model of $\Sigma \wedge F$.

On the other hand, if G_Σ is a linear chain and $v(l)$ is in the first clause $(v(l) \in v(c_1))$ but it is not the label of (α_1, β_1). Supposing $F = \{(l^\epsilon)\}, \epsilon \in \{0,1\}$, then we determine the pair (α_1, β_1), as: $(\alpha_1, \beta_1) = \begin{cases} (1,1) \text{ if } \epsilon = \epsilon_1 \\ (1,0) \text{ if } \delta_1 = 1 \\ (0,1) \text{ if } \delta_1 = 0 \end{cases}$

With this modification, the application of the recurrence (2) continues until we obtain the last pair (α_m, β_m) and we proceed as was described in (iii) or (iv) according that G_Σ being a linear chain or a cycle.

Notice that if $\Sigma \wedge F$ is unsatisfiable then $P_{F|\Sigma} = 0$. Since we are not updating Σ with the information of F, we do not have to update or revise the knowledge base Σ, which is the most common procedure when regarding update and belief revision, and then $P_{F|\Sigma}$ continues being well-defined.

If $F = \{(l)\}$ and $v(l) \notin v(\Sigma)$, and as we have considered the degree of belief in F given Σ as a conditional probability, then it makes sense to *update* the degree of belief for *updating* the probability space where $P_{F|\Sigma}$ is computed [6].

4.1 Updating Probabilities and Degrees of Belief

When new pieces of information that did not originally appear in the sample space must be considered, we will introduce the area of updating the degree of belief for making an extension of the original probability space [4].

Let $F = (\bigwedge_{j=1}^{k} l_j)$ a conjunction of literals where there are variables that do not appear in the original knowledge base Σ. Let $A = \{l \in F : v(l) \notin v(\Sigma)\}$, $t = |A|$ and $|v(\Sigma)| = n$. We consider F as a set of literals (the conjunction is understood between the elements of the set), let $F' = F - A$. There are 2^n assignments defined over $v(\Sigma)$ and 2^{n+t} assignments defined over $v(\Sigma) \cup v(F)$, then we *update* the domain of the probability space over which we compute $P_{F|\Sigma}$, as: $P_{F|\Sigma} = \frac{Prob_{(\Sigma \wedge F)}}{Prob_{\Sigma}} = \frac{\frac{\mu(\Sigma \wedge F)}{2^{n+t}}}{\frac{\mu(\Sigma)}{2^n}} = \frac{\mu(\Sigma \wedge F)}{2^t \cdot \mu(\Sigma)}$. Since G_A and $G_{\Sigma \cup F'}$ are two independent connected components and $\mu(\bigwedge_{l \in A} l) = \prod_{l \in A} \mu(l) = 1$, then:

$$P_{F|\Sigma} = \frac{\mu(\Sigma \wedge F') \cdot \mu(\bigwedge_{l \in A} l)}{2^t \cdot \mu(\Sigma)} = \frac{\mu(\Sigma \wedge F')}{2^t \cdot \mu(\Sigma)} \tag{3}$$

Example 3. *Let* $\Sigma = \{(x_0, x_1), (x_1, x_2), (x_2, x_3), (x_3, x_4), (x_4, x_5)\}$*, the formula from the Example 1 and* $F = \{x_0, \overline{x}_3, \overline{x}_6\}$. $A = \{l \in F : v(l) \notin v(\Sigma)\} = \{\overline{x}_6\}$*, and* $t = \mid A \mid = 1$*. As* x_0 *appears with the same sign in both F and the tail of the chain* G_Σ*,* $(\alpha_1, \beta_1) = (1, 1)$. $(\alpha_2, \beta_2) = (\alpha_1 + \beta_1, \alpha_1) = (2, 1) \rightarrow (3, 2) = (\alpha_3, \beta_3)$*, but this last pair must be changed since* $v(x_3)$ *is the label of* (α_3, β_3) *and appears as a negated variable in F, then* $(\alpha_3, \beta_3) = (0, \beta_3) = (0, 2)$. $(\alpha_4, \beta_4) = (\alpha_3 + \beta_3, \alpha_3) = (2, 0) \rightarrow (2, 2) = (\alpha_5, \beta_5)$*. And* $\mu_5 = 4 = \mu(\Sigma \wedge F) = \mu(\Sigma \wedge F') = \mu(\Sigma \wedge (x_0) \wedge (\overline{x}_3))$*, according to (3) since* x_6 *does not appear in* $v(\Sigma)$. $P_{F|\Sigma} = \frac{\mu(\Sigma \wedge \bigwedge_{j=1}^{k} l_j)}{2^t \cdot \mu(\Sigma)} = \frac{\mu(\Sigma \wedge (x_0) \wedge (\overline{x}_3) \wedge (\overline{x}_6))}{2^t \cdot \mu(\Sigma)} = \frac{4}{2 \cdot 21} = \frac{4}{42}$.

Proposition 1. *Let* Σ *a* $(\leq 2, 3\mu)$*-CF, where* Σ *can be decomposed into two subformulas:* $\Sigma = \Sigma_1 \cup \Sigma_2$ *such that* Σ_1 *is a* $(2, 2\mu)$*-CF and* Σ_2 *is a* $(1, 1\mu)$*-CF, then* $\mu(\Sigma)$ *can be computed in polynomial time.*

Proof: Suppose (without a loss of generality) that $\Sigma_2 = (\bigwedge_{j=1}^{k}(l_j))$. Then $\mu(\Sigma) = \mu(\Sigma_1 \cup \Sigma_2) = \mu(\Sigma_1 \wedge \bigwedge_{j=1}^{k} l_j)$. And we have shown (based on the Equation (3)) how to efficiently compute $\mu(\Sigma_1 \wedge \Sigma_2)$ where Σ_1 is a $(2, 2\mu)$-CF and Σ_2 is a conjunction of unitary clauses.

Corollary 1. *There is a subclass of formulas in* $(\leq 2, 3\mu)$*-CF which are not in* $(\leq 2, 2\mu)$*-CF and where the #SAT problem can be computed in polynomial time.*

It is easy to determine if Σ can be decomposed as the above proposition expresses, because we must merely split the clauses from Σ into two sets, one of which contains binary clauses and the other contains only unitary clauses.

Let now a clause, $F = (\bigvee_{j=1}^{k} l_j)$. Let $A = \{l \in F | v(l) \notin v(\Sigma)\}$, and $t = \mid A \mid$. Considering F as a set of literals, let $F' = F - A$. We can compute $\mu(\Sigma \wedge F)$, as: $\mu(\Sigma \wedge F) = \mu(\Sigma) \cdot 2^t - \mu(\Sigma \wedge \overline{F})$. Then, we are computing $\mu(\Sigma \wedge F)$ by extending the models of Σ for considering the variables which are in $v(F)$ however they are not in $v(\Sigma)$, and we are eliminating the assignments which falsify $\Sigma \cup F$.

As, $F = (\bigvee_{j=1}^{k} l_j)$ then $\overline{F} = (\bigwedge_{j=1}^{k} \overline{l}_j) = (\bigwedge_{x \in F'} \overline{x} \wedge \bigwedge_{x \in A} \overline{x})$ since $v(A) \cap (v(\Sigma) \cup v(F')) = \emptyset$ we could consider A a connected component independent to $G_{\Sigma \cup F'}$, where $\overline{F'} = \bigwedge_{x \in F \wedge v(x) \in v(\Sigma)} \overline{x}$. According to (1), $\mu(\Sigma \wedge \overline{F}) = \mu(\Sigma \wedge \overline{F'}) \cdot \mu(\overline{A}) = \mu(\Sigma \wedge \overline{F'})$ since $\mu(\overline{A}) = 1$, then:

$$P_{F|\Sigma} = \frac{\mu(\Sigma \wedge F)}{2^t \cdot \mu(\Sigma)} = \frac{\mu(\Sigma) \cdot 2^t - \mu(\Sigma \wedge \overline{F'})}{2^t \cdot \mu(\Sigma)} = 1 - \frac{\mu(\Sigma \wedge \overline{F'})}{2^t \cdot \mu(\Sigma)} \tag{4}$$

Notice that when F is a phrase or a clause there is no restriction on the number of literals that F can contain. Thus, (4) permits us to solve #SAT forformulas $(\Sigma \cup F)$ in a greater hierarchy than $(\leq 2, 3\mu)$-CF, for considering clauses in F with more than 2 literals.

Example 4. *Let Σ be the formula from Example 1, and $F = \{(x_0 \vee x_5)\}$. What is the value of $\mu(\Sigma \wedge F)$?. Notice that $\Sigma \cup F$ conforms a cycle, and $\mu(\Sigma \wedge F)$ is the number of models in such cycle. $A = \{x \in F | v(x) \notin v(\Sigma)\} = \emptyset$ and $F' = F$. Then $\mu(\Sigma \wedge F) = \mu(\Sigma) \cdot 2^0 - \mu(\Sigma \wedge \overline{F}) = \mu(\Sigma) - \mu(\Sigma \wedge (\overline{x}_0) \wedge (\overline{x}_5))$. $\mu(\Sigma \wedge (\overline{x}_0) \wedge (\overline{x}_5)) = \mu_5$, computed in the next way: As x_0 is the tail of the chain in G_Σ and has a different sign in F and Σ then $(\alpha_1, \beta_1) = (1, 0)$, since $\delta_1 = 1$. After,* (1, 0) $\to$ (1,1) $\to$ (2,1) $\to$ (3, 2) $\to$ (5, 3) *and as x_5 also appears as negated variable in F then $\mu_5 = \beta_5 = 3$. Finally, $P_{F|\Sigma} = 1 - \frac{\mu(\Sigma \wedge \overline{F'})}{2^0 \cdot \mu(\Sigma)} = 1 - \frac{3}{21} = \frac{18}{21}$. This example shows another way to compute $\mu(\Sigma)$, when G_Σ is a cycle.*

Some methods for choosing among several possible revisions is based on some implict bias, namely the a priory probability that each element (literal or clause) of the domain theory requires revision. Opposite to assign the probabilities to each element F of the theory Σ by an expert or simply chosen by default [7], we have shown here a formal and efficient way to determine such probability based on the degree of belief $P_{F|\Sigma}$, with the additional advantages that such probabilities could be adjusted automatically in response to newly-obtained information.

4.2 A Polynomial Superclass of $(2, 2\mu)$-CF for #SAT

Let now F a $(2, 1\mu)$-CF. Let $A = \{l \in F | v(l) \notin v(\Sigma)\}$, and $t = | A |$. Let $F = F_1 \cup F_2$ where $F_2 = \{c \in F \mid v(c) \cap v(\Sigma) = \emptyset\}$, and $F_1 = F - F_2$. We compute $\mu(\Sigma \wedge F)$ as $\mu(\Sigma \wedge (F_1 \cup F_2)) = \mu(\Sigma \wedge F_1) \cdot \mu(F_2)$ since G_{F_2} is a connected component independent to $G_{\Sigma \cup F_1}$. Considering $n_1 = | v(F_2) |$ then $\mu(F_2) = 3^{\frac{n_1}{2}}$.

We order the clauses in Σ, as: $\Sigma = \left\{c_i = \{y_{i-1}^{\epsilon_i}, y_i^{\delta_i}\}\right\}_{i=1}^{m}$ $\delta_i, \epsilon_i \in \{0,1\}$ and $v(c_i) \cap v(c_{i+1}) = \{y_i\}$. According to this order, we order the clauses of $F_1 = \{D_1, ..., D_{m_1}\}$. For any two clauses $D_i = (l_1, l_2)$ and $D_j = (l_3, l_4)$, $i < j$: if $v(l_1) \in v(\Sigma)$ then $v(l_1)$ appears in a clause of Σ before any other clause where the variables $v(l_2)$, $v(l_3)$ or $v(l_4)$ appear. If $v(l_1) \notin v(\Sigma)$ then $v(l_2)$ appears in a clause of Σ before any other clause where the variables $v(l_3)$ or $v(l_4)$ appear.

In order to compute $\mu(\Sigma \wedge F_1)$, meanwhile we have applied (2) for each $i = 1, ..., m$ in order to obtain the three pairs: $(\alpha_i, \beta_i)(s_j), j = 1, 2, 3$, we determine if the associated label y_i is in $v(F_1)$ or not.

If $y_i \notin v(F_1)$ no changes are applied to $(\alpha_i, \beta_i)(s_j), j = 1, 2, 3$, and we continue to compute the next values: $(\alpha_{i+1}, \beta_{i+1})(s_j), j = 1, 2, 3$.

If $y_i \in v(F_1)$, the obtained values $(\alpha_i, \beta_i)(s_j), j = 1, 2, 3$ have to be changed. And for explaining this change, suppose $D = (y_i^\epsilon, z^\delta) \in F_1$, $\epsilon, \delta \in \{0, 1\}$, then:

1. $z \notin v(\Sigma)$ then each $(\alpha_i, \beta_i)(s_j), j = 1,2,3$ is modified according to the sign of y_i in D, as: $(\alpha_i, \beta_i)(s_j) = \begin{cases} (2 \cdot \alpha_i, \beta_i)(s_j) & \text{if } \epsilon = 1, \\ (\alpha_i, 2 \cdot \beta_i)(s_j) & \text{Otherwise.} \end{cases}$
2. $z \in v(\Sigma)$ and a clause $c \in \Sigma$ exists such that $c = (y_i^{\epsilon'}, z^{\delta'})$, $\epsilon', \delta' \in \{0,1\}$ if $\epsilon = \epsilon'$ and $\delta = \delta'$ then D and c are the same clause, so we do not modify (α_i, β_i) and continue to compute the next pair, if $\epsilon \neq \epsilon'$ and $\delta \neq \delta'$ then y_i and z are complementary literals in D and c, and we compute the next pairs of values, as: $(\alpha_{i+1}, \beta_{i+1})(s_j) = \begin{cases} (\alpha_i, \beta_i)(s_j) & \text{if } \epsilon \neq \delta, \\ (\beta_i, \alpha_i)(s_j) & \text{Otherwise.} \end{cases}$
   In other case, only one of the two literals y_i or z changes its sign, then:
   $$(\alpha_{i+1}, \beta_{i+1})(s_j) = \begin{cases} (\alpha_i + \beta_i, 0)(s_j) \text{ if } \delta = \delta' = 1 \\ (0, \alpha_i + \beta_i)(s_j) \text{ if } \delta = \delta' = 0 \\ (2 \cdot \alpha_i, 0)(s_j) \text{ if } \epsilon = \epsilon' = 1 \\ (0, 2 \cdot \beta_i)(s_j) \text{ if } \epsilon = \epsilon' = 0 \end{cases}$$
3. $z \in v(\Sigma)$ and the variables: y_i, z appear in different clauses of Σ. Let $D = (y_i^{\epsilon}, z^{\delta}) \in F_1$ and $c_i = (y_{i-1}^{\epsilon_i}, y_i^{\delta_i})$, $c_k = (y_{k-1}^{\epsilon_k}, z^{\delta_k})$ the two clauses of Σ, with $z = y_k$. Notice that $i < k$ since the order given over the clauses of Σ and of F_1. We begin a new series of values $(\alpha_l^1, \beta_l^1)(s_j), j = 1,2,3, l = i, \ldots, k$, where the initial pair is: $(\alpha_i^1, \beta_i^1)(s_j) = \begin{cases} (\alpha_i, 0)(s_j) & \text{if } \epsilon = 0, \\ (0, \beta_i)(s_j) & \text{if } \epsilon = 1. \end{cases}$

   The computation of the new series $(\alpha_l^1, \beta_l^1)(s_j), j = 1,2,3$ (new level) continues in a parallel way of the computation of the main series $(\alpha_i, \beta_i)(s_j), j = 1,2,3, i = 1, \ldots, m$ until we arrive to the pair (α_k, β_k) which has associated the label y_k. All arithmetic rules applied to the main series (α_i, β_i) in order to obtain the next three pairs $(\alpha_{i+1}, \beta_{i+1})$ must be applied to (α_l^i, β_l^i) in order to obtain the next pairs $(\alpha_{i+1}^1, \beta_{i+1}^1)$.

   Notice that more levels of series $(\alpha_i^{(l)}, \beta_i^{(l)})(s_j), j = 1,2,3$ could be needed to compute and which are embedded to the main series $(\alpha_i, \beta_i)(s_j), j = 1,2,3$.
   The new series $(\alpha_l^1, \beta_l^1)(s_j), j = 1,2,3,\ l = i, ..., k$ stops being active when we arrive to the clause $c_k \in \Sigma$. According to the sign of $z \in v(F_1)$ we obtain the last pair, as: $(\alpha_k^1, \beta_k^1)(s_j) = \begin{cases} (\alpha_k^1, 0)(s_j) & \text{if } \delta = 1, \\ (0, \beta_k^1)(s_j) & \text{Otherwise.} \end{cases}$

   These last three pairs are used to modify the values in (α_k, β_k) of the original series and in whatever pairs $(\alpha_k^{(l)}, \beta_k^{(l)})$ that could be active in that moment and such that $\alpha_i^{(l)}(s_j) \leq \alpha_i^1(s_j)$ and $\beta_i^{(l)}(s_j) \leq \beta_i^1(s_j), j = 1,2,3$. The new values are obtained, as: $(\alpha_k^{(l)}, \beta_k^{(l)})(s_j) = (Max\{0, \alpha_k^{(l)} - \alpha_k^1\}, Max\{0, \beta_k^{(l)} - \beta_k^1\})(s_j)$. Since the set of assignments denoted by (α_k^1, β_k^1) are being considered in the set of assignments denoted by $(\alpha_k^{(l)}, \beta_k^{(l)})$ and in (α_k, β_k).

   All computation levels of the series $(\alpha_i^{(l)}, \beta_i^{(l)})(s_j), j = 1,2,3$ will end before arriving to the last pair $(\alpha_m, \beta_m)(s_j), j = 1,2,3$ in the main series, and according to (iv), $\mu(\Sigma \cup F_1) = t(s_1) + t(s_2) + t(s_3)$.

The above procedure permits us to compute $\mu(\Sigma \wedge F_1)$, then we can apply (3) in order to compute $P_{F|\Sigma}$, as: $P_{F|\Sigma} = \frac{\mu(\Sigma \cup F_1)\cdot\mu(F_2)}{2^{n_1}\cdot\mu(\Sigma)} = \frac{\mu(\Sigma \wedge F_1)\cdot 3^{n_1/2}}{2^{n_1}\cdot\mu(\Sigma)}$. This last equation justifies the following proposition.

Proposition 2. *There is a subclass form by $\Sigma_1 \cup \Sigma_2$ of the class $(2, 3\mu)$-CF which is not in the class $(2, 2\mu)$-CF and such that the #SAT problem can be computed in polynomial time. Being Σ_1 a connected component in $(2, 2\mu)$-CF and Σ_2 a $(2, 1\mu)$-CF.*

On the other hand, if we do not put any restrictions on each of the formulas Σ and F such that $\Sigma \cup F$ is a $(2, 3\mu)$-CF, then both the #SAT problem and the degree of belief $P_{F|\Sigma}$ are #P-complete problems.

5 Conclusions

We determined the degree of belief $P_{F|\Sigma}$ based on the conditional probability of F given Σ. Given an initial knowledge base Σ as a $(2, 2\mu)$-CF, we designed efficient procedures to compute $P_{F|\Sigma}$ when F is a conjunction of unitary clauses or F is a disjunction of unitary clauses or F is a $(2, 1\mu)$-CF.

To consider the degree of belief as a conditional probability looks promising for determining more polynomial classes of propositional formulas for computing the degree of belief as well as for the #SAT problem. Although #SAT is a #P-complete problem for formulas in $(\leq 2, 3\mu)$-CF without any restriction, we have shown here a class of formulas given by $(\Sigma \cup F)$ which is in a greater hierarchy than $(\leq 2, 3\mu)$-CF and where both computing the degree of belief $P_{F|\Sigma}$ and solving #SAT over $(\Sigma \cup F)$ can be done in polynomial time.

Also, we have established new polynomial classes of conjunctive forms which are subclasses of $(2, 3\mu)$-CF and superclasses of $(2, 2\mu)$-CF for the #SAT problem. Each one of those classes can be described as the union of two formulas $F \cup \Sigma$ and such that $P_{F|\Sigma}$ is computed efficiently.

It is an open problem to determine the maximal polynomial subclass Γ of $(2,3\mu)$-CF for the #SAT problem. As well as to know if such Γ could always be described as the union of two formulas $F \cup \Sigma$ where $P_{F|\Sigma}$ can be computed efficiently.

References

1. Bayardo R. Jr., Pehoushek J.D., Counting Models using Connected Components, *Proceeding of the Seventeenth National Conf. on Artificial Intelligence,* 2000.
2. Darwiche Adnan, On the Tractability of Counting Theory Models and its Application to Belief Revision and Truth Maintenance, *Jour. of Applied Non-classical Logics,* 11(1-2), (2001), 11-34.
3. Eiter T., Gottlob G., The complexity of logic-based abduction, *Journal of the ACM 42(1),* (1995), 3-42.
4. Fagin R., Halpern J. Y., *A new approach to updating beliefs,* Uncertainty in Artificial Intelligence 6, eds. P.P. Bonissone, M. Henrion, L.N. Kanal, J.F. Lemmer, (1991), 347-374. *Artificial Intelligence 54,* (1992), 275-317.

5. Goran Gogic, Christos H. Papadimitriou, Marta Sideri, Incremental Recompilation of Knowledge, *Journal of Artificial Intelligence Research 8,* (1998), 23-37.
6. Halpern J. Y., Two views of belief: Belief as generalized probability and belief as evidence, *Artificial Intelligence 54,* (1992), 275-317.
7. Koppel M., Feldman R., Maria Segre A., Bias-Driven Revision of Logical Domain Theories, *Jour. of Artificial Intelligence Research 1,* (1994), 159-208.
8. Roth D., On the hardness of approximate reasoning, *Artificial Intelligence 82,* (1996), 273-302.
9. Russ B., *Randomized Algorithms: Approximation, Generation, and Counting,* Distingished dissertations Springer, 2001.
10. Zanuttini B., New Polynomial Classes for Logic-Based Abduction, *Journal of Artificial Intelligence Reseach 19,* (2003), 1-10.

Combining Quality Measures to Identify Interesting Association Rules

Edson Augusto Melanda[1,2] and Solange Oliveira Rezende[2]

[1] Federal University at São Carlos – Department of Civil Engineering, Nucleus of Geo-processing – NGeo – São Carlos, SP, Brazil
melanda@power.ufscar.br

[2] University of São Paulo – Institute of Mathematics and Computer Science, Department of Computer Science and Statistics – São Carlos, SP, Brazil
solange@icmc.usp.br

Abstract. Association rules is a descriptive data mining technique, that has acquired major interest in the last years, with applications in several areas, such as: electronic and traditional commerce, securities, health care and geo-processing. This technique allows identifying intra-transactions patterns in a database. An association rule describes how much the presence of a set of attributes in a database's record implicates in the presence of other distinct set of attributes in the same record. The possibility of discovering all the existent associations in a database transaction is the most relevant aspect of the technique. However, this characteristic determines the generation of a large number of rules, hindering the capacity of a human user in analyzing and interpreting the extracted knowledge. The use of rule measures is important to analyze the knowledge. In this context we investigate, firstly, the intensity of several objective measures to act as filters for rule sets. Next, we analyze how the combination of these measures can be used to identify the more interesting rules. Finally, we apply the proposed technique to a rule set, to illustrate its use in the post-processing phase.

Keywords: Association Rules, Post-processing, Objective Measures.

1 Introduction

Technological advances in data acquisition and storage systems, together with increasing in communication speed and reduction of costs associated with these technologies have provided to the organizations the capacity to store a detailed set of information about their operations, generating a considerable amount of data. At the same time, the value of the information contained in this data has been acknowledged by the organizations. This scenery has stimulated research with the purpose of automating the process of transforming data into knowledge. This process of automatic identification of knowledge is known as data mining.

Among data mining techniques, association rules has been among others, the one that has aroused more interest [1]. In the academic area, some researches have been developed and a lot of organizations have been using this technique thoroughly, in applications in areas such as commerce, securities, health, geoprocessing [2,3].

In general, the association rules technique describes how much the presence of a set of attributes in a database's record implicates in the presence of other distinct set of attributes in the same record. The possibility of finding all existent associations is the most important aspect of the technique. However, this characteristic determines the generation of a huge number of rules, hindering the interpretation of the rules set for the user. To deal with this problem, techniques of post-processing knowledge, that allow the identification of interesting rules, have been the object of several studies. A methodology and a tool for navigation and visualization of rules are proposed in [4]. This tool is based on a set of operators that make possible to focus on specific characteristics of the rules set, considering different aspects. A framework that combines statistical and graphical techniques for pruning rules is proposed in [5]. A two-step methodology for rule selection is proposed in [6], on the first step the statistical measure chi-square is used and then, objective measures of interestingness.

In this context, we investigate the potentiality of objective measures derived from contingency table to be used as a filter to identify interesting rules. The main idea is to identify measures that follow Pareto's Law. This law states that for many phenomena, 80% of consequences stem from 20% of the causes. Applying this principle in the filtering of the associating rules, it's possible to prioritize the set of rules to present to a specialist.

This investigation is carried out through the analysis of the graphical form of the mathematical function of each measure and its stability for distinct databases. Using manual evaluation, we grouped the graphics by their graphical form. This produced four groups of functions: linear decline, accentuated fall after a constant landing, sigmoidal and accentuated initial decline. In order to a measure be used as a filter, it's desirable that this measurement follows the Pareto principle (in the graph, this is represented by an accentuated initial decline). It is also desirable, that this graphical form keeps its characteristics independently of the databases and the parameters used in the data mining processes. In the sequence, we analyze how combinations of these the measures can be used to reduce further more the dimensionality of the rules set.

2 Background

The association rules was undertaken firstly for use in binary databases in [7]. In subsequent articles – [8, 9], among others – this approach was enlarged and widespread, considering aspects such as continuous attributes, attributes taxonomy, relational and distributed databases, and parallel processing.

Association Rule mining is commonly stated as follows [7]: Let $I = \{i_1, \ldots, i_n\}$ be a set of *items,* and D be a set of data cases. Each data case consists of a subset of items in I. An association rule is an implication of the form $LHS \rightarrow RHS$, where $LHS \subset I$, $RHS \subset I$, and $LHS \cap RHS = \oslash$. The rule $LHS \rightarrow RHS$ has support s in D if $s\%$ of the data case in D contains $LHS \cup RHS$ (Equation 1).

The rule holds in D with confidence c if $c\%$ of data cases in D that supports *LHS* also support *RHS* (Equation 2). *LHS* and *RHS* are, respectively, the left and right hand side of a rule. The problem of mining association rules is to generate all association rules that have support and confidence greater than the user-specified minimum support and minimum confidence.

$$\text{support} = f(LHS\ RHS) \tag{1}$$

$$\text{confidence} = f(RHS \mid LHS) = \frac{f(LHS\ RHS)}{f(LHS)} \tag{2}$$

The measures support and confidence are the most used in association rules, either in the step of attributes subsets selection, during the process of rule generation, or in the post-processing phase of the acquired knowledge.

Besides support and confidence, other measures for knowledge evaluation have been researched with the purpose of supplying subsidies to the user in the understanding and use of the acquired knowledge [10, 6, 11, 9]. These metrics are defined in terms of the frequency counts tabulated in a 2 X 2 contingency table as shown in Table 1.

Table 1. A contingency table for a rule $LHS \rightarrow RHS$

	RHS	$\overline{RHS}$	
LHS	f($LHS\ RHS$)	f($LHS\ \overline{RHS}$)	f(LHS)
$\overline{LHS}$	f($\overline{LHS}\ RHS$)	f($\overline{LHS}\ \overline{RHS}$)	f($\overline{LHS}$)
	f(RHS)	f($\overline{RHS}$)	N

Measure relative specificity that represents the specificity gain in relation to $LHS \rightarrow \text{true}$. The measure relative specificity (Equation 3) is defined by the expression:

$$\text{relative specificity} = f(\overline{RHS} \mid \overline{LHS}) - f(\overline{LHS}) \tag{3}$$

Measure J_1 (Equation 4) represents the relationship between generality and discriminat capacity of a rule, when it is considered the discrimination capacity in the cases in that $f(RHS \mid LHS)$ is greater than $f(RHS)$[9].

$$J_1 = f(LHS) \cdot f(RHS \mid LHS) \cdot \log_2 \frac{f(RHS \mid LHS)}{f(RHS)} \tag{4}$$

Presented metric are usually used in association rules post-processing to sort rules. Using Pareto's analysis it is possible, besides ordering the rules, to establish a method to filter these rules. Pareto's analysis is a commonly used method of separating the major causes (the "vital few") of a problem, from the minor ones (the "trivial many"). It helps prioritize and focus resources where they are most needed by showing where initial effort should be placed to produce the most gain. It also helps measure the impact of an improvement by comparing before and after conditions.

3 Analysis of Objective Measures

Experiments with ten databases and thirteen objective measures aimed at analyzing the behavior of the functions represented by objective measures for a possible use into the knowledge post-processing phase were carried out. A complete description of the experiment, including the databases, the appliances and the parameters used in the process of data mining, and its results are available in [12].

3.1 Used Databases and Applications

The databases used in the experiment, two are of real data – about urban quality of life (UQL) and about electronic commerce (BMS-POS) – and eight are artificial databases. UQL database is composed by the set of variables of life quality considered in the formulation of Indicator of the Urban Quality of Life and the own index, regarding urban sectors of the city of São Carlos, SP, Brazil. BMS-POS[1] is a database that contains transactions of electronic commerce web site, for a period of two months and it was used at KDD-cup 2000 [13]. The attributes are related to three categories of information: click streams, order information and registration form.

The artificial databases in our experiments (called of Art A, Art B, Art C, etc.) were generated using the application Gen, which produces artificial bases from synthetic data, from user-specified parameters. This application is available at `www.almaden.ibm.com/cs/quest/syndata.html` and it is used broadly in tests of association rules algorithms. The used attributes are of the discreet type, given the specific characteristics of the association rules problem.

Characteristics of all databases are described in Table 2, including the average number of attributes for example (considering a database of market shopping, represents the average number of bought items) and the maximum number of attributes by rule that will limit the size of the rule.

Table 2. Characteristics of the databases and its results from the data mining process

Database	Number of attributes	Number of examples	Average of attributes per example	Minimum Support	Minimum Confidence	Maximum attributes per rule	Number of rules
Art A	1,000	1,000	25	6	25	5	44
Art B	10	25,000	5	6	25	5	2,000
Art C	10	25,000	9	6	25	5	2,500
Art $D_{(2)}$	100	25,000	80	6	25	2	9,000
Art $D_{(3)}$	100	25,000	80	6	25	3	440,000
Art $D_{(4)}$	100	25,000	80	6	25	4	600,000
Art E	1,000	1,000	100	6	25	5	20,000
Art F	19	25,000	25	6	25	5	80,000
Art G	30	25,000	25	6	25	5	420,000
Art H	30	2,500	25	6	25	5	430,000
UQL	119	120	119	2	50	3	24,713
BMS-POS	1,657	515,597	–	2	25	5	70,000

[1] We wish to thank Blue Martini Software for contributing the BMS-POS database.

Databases were mined varying the minimum values of support and confidence. To verify the influence of another parameter, database Art D was mined varying the maximum number of attributes per rule. The used values were 2, 3 and 4 attributes per rule (the default value for this parameter is 5). Default values were used to other parameters. The mining process result is also displayed in Table 2.

3.2 Graphical Analysis of the Measures

In order to analyze the behavior of the functions, each measure was arranged individually in a decreasing way and the values plotted in graphs, with the values of the measures in the ordinates and the sequential number of the rule in the abscissas. This procedure has produced 182 graphs – one graph for each measure and for each execution of the data mining algorithm.

The first step in the analysis process was to identify classes of functions with similar behavior. It was made by classifying manually each graphic. It was decided to analyze the graphs similarity, instead of analyzing equations likeness, since we want to analyze a more generic similarity, related mainly to the decline of the curve. Another aspect considered was the size of some databases that would difficult the identification of the equations.

In the Table 3 are presented the results of the generated classification. This makes possible to verify the stability of the graphic behavior in relation to databases and to the mining parameters.

When analyzing data contained in Table 3, it was observed that the measures relative negative confidence and relative sensibility are the ones that present the smallest variability, followed by the measures relative specificity, specificity and support. This indicates that those measures could be used as filters in a system that lacks user's supervision. For the other measures it would be necessary more user's supervision, as the case of the measure confidence, that holds high variability in the different databases.

Category I contains the group of graphs with functions that have accentuated initial decline, and that is the most interesting format for use as a filter since this format is closer to the one of the Pareto distribution. Categories L and C contain the groups of graphs with functions that have linear decline and decline after a constant landing, respectively. Functions of this type do not follow the Pareto's distribution, thus they have low restrictive capacity and are little useful as filters. The S category groups function graphs, which have the sigmoidal format. Functions of this kind can be useful for analysis in localized regions of the curve (beginning and ending), since the intermediary part is linear and offers little action as a filter.

Another aspect to be considered is that the measures present variation on the graph format, that determines the need to verify the measure graphic format for each base of rule.

These types of graphic behavior can be observed in Figures 1 to 3, that were selected for their representativeness in the carried out study.

Table 3. Graphs classified in categories

Measure	A	B	C	$D_{(2)}$	E	UQL	BMS-POS	H	I	J	$D_{(3)}$	$D_{(4)}$	Mode
confidence	L	C	C	C	I	C	C	C	L	C	C	C	C
conviction	I	I	I	I	C	I	I	I	I	I	L	L	I
negative confidence	L	S	S	S	I	S	L	S	S	S	I	I	S
novelty	C	I	S	S	I	I	I	S	S	S	I	I	I
relative confidence	L	I	S	I	I	I	L	I	S	S	I	I	I
relative negative confidence	S	I	I	I	I	I	I	I	I	I	I	I	I
relative sensibility	L	I	I	I	I	I	I	I	I	I	I	I	I
relative specificity	S	I	S	I	I	I	I	S	I	I	I	I	I
satisfaction	L	I	S	I	I	I	C	L	S	S	I	I	I
sensibility	I	I	I	L	C	I	I	L	I	I	L	L	I
specificity	C	C	C	L	C	C	C	C	C	C	L	L	C
support	I	I	I	I	L	I	I	I	I	I	L	L	I
J_1	S	I	S	S	I	I	I	S	S	S	I	I	I/S

Categories

I	Initial accentuated decreasing	S	Sigmoidal
L	Linear decreasing	C	Decreasing after landing

In the graphs of the confidence measure (Figure 1), can be observed the occurrence of an initial portion approximately constant and an accentuated decrease only after 50% of the total number of rules (Figures 1(a) and 1(c)). Usually measures with these characteristics are not useful as filters. However, they can be used in other types of analyzes or as a "pre-filter". In addition, it is observed that the graph of Figure 1(b) distinguishes considerably of the others. This indicates instability in the graphical form of the measure, what also disqualifies the measure for use as isolated filter.

In the graphs of the support measure (Figure 2), it is possible to identify an initial accentuated decrease of the values, especially in the Figures 2(a) and 2(b). Analyzing the Figure 2(a) in details, it was observed that for the first 750 rules (value below to 5% of the total of rules), the value of the measure support varies in 96% (from 0.75 to 0.05). For the other rules, the value of support varies from 0.05 to 0.02 (minimum support). This condition makes possible the identification of a reduced number of rules, and this measure can be used as a filter.

In relation to the measure graphic format stability, it is identified a quite similar behavior among the bases. However for the database Art D (Figure 2(c)), the function decreases almost linearly until reaching the asymptote. Considering this small instability, the measure is indicated as a filter, with the user's supervision. Measures with similar characteristics can also be used as filters, with the same restrictions for this measure.

The relative specificity measure, Figure 3, presents an accentuated initial decrease, on all databases considered in the study. Like this, the use of this measure is indicated as filter for rule selection, once attending the need of accentuated initial decrease, the graphical behavior keeps constant for distinct databases.

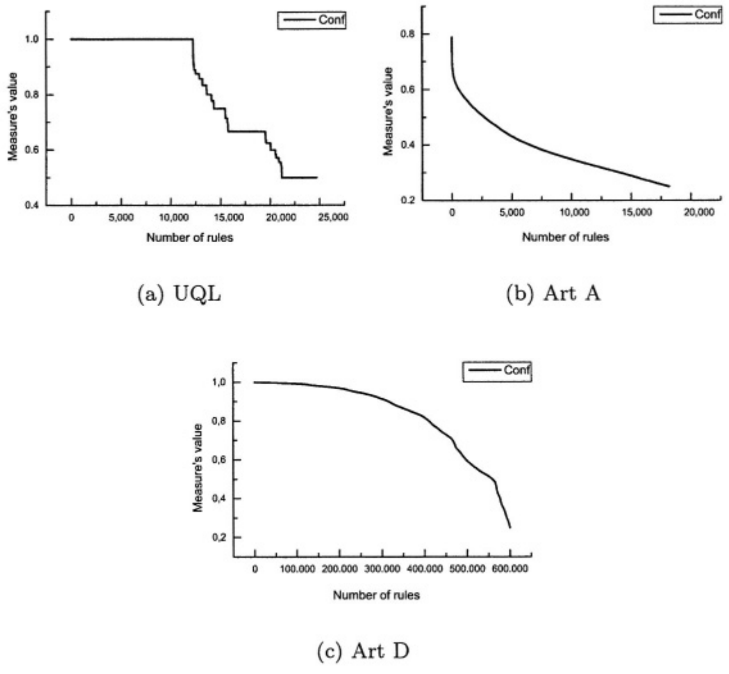


Fig. 1. Behavior of the confidence measure

Observing in details the Figure 3(c), it is verified that altering the minimum value of relative specificity from 0.0 to 0.1 (decrease of 20%), the number of rules reduces in approximately 82%, that demonstrates the filter capacity. Measures with similar characteristics can also be used as filters, with or without the user's supervision.

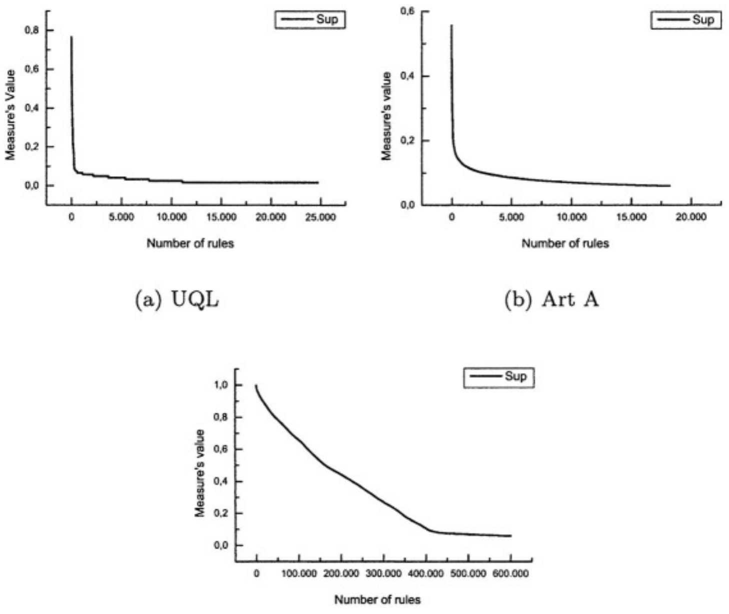
It is valid to say that a measure, adopted in an isolated way, can usually supply few subsidies for a valuable analysis of the rules sets. But its combined use, for example through the composition of filters, can lead to, even more useful results.

4 Using Measures as Rules Filters

In this section, we evaluate the measures use as an individual or a compound filter, being considered fundamentally the reduction of the number of rules. For this evaluation, the UQL database was selected, because is possible to interpret the rules generated from this database.

The measures confidence, J_1, relative specificity and support were selected because they represent different graphs formats. The analysis of the restrictive power for each measure is accomplished through the elevation of its minimum value.

On Table 4 is shown the results of the use of each measure as an individual filter. In general, the measures support and relative specificity are the most



(a) UQL

(b) Art A

(c) Art D

Fig. 2. Behavior of the support measure

restrictive. It is verified that a elevation of only 5% in the minimum value of each measure leads to a reduction of approximately 90% in the number of rules, and for a elevation of 10% in the measure's value, the number of rules is reaching about 99% of reduction. Being this, these measures can be considered as very restrictive filters. In terms of Pareto principles, there is that, for 95% of the consequences (5% of elevation), we have 10% of causes, and for 90% of the consequences (10% of elevation) are produced by only 1% of causes.

Table 4. Number of rules for each percentage of elevation in the measure's minimum value

	Percentage of elevation in the measure's minimum value				
	5%	10%	30%	50%	80%
Measure	Number of selected rules (percentage of reduction in the number of rules)				
Confidence	21,169 (14.34)	21,095 (14.64)	19,526 (20.99)	15,467 (37.41)	12,360 (49.99)
Support	2,137 (91.35)	303 (98.77)	131 (99.47)	74 (99.70)	2 (99.99)
Relative specificity	2,888 (88.31)	234 (99.05)	57 (99.77)	11 (99.96)	2 (99.99)
J_1	16,139 (34.69)	11,143 (54.91)	1,414 (94.28)	296 (98.80)	12 (99.95)

The J_1 measure can be considered as a rather restrictive filter, because only with reduction greater than 50%, occurs expressive decrease in the number of

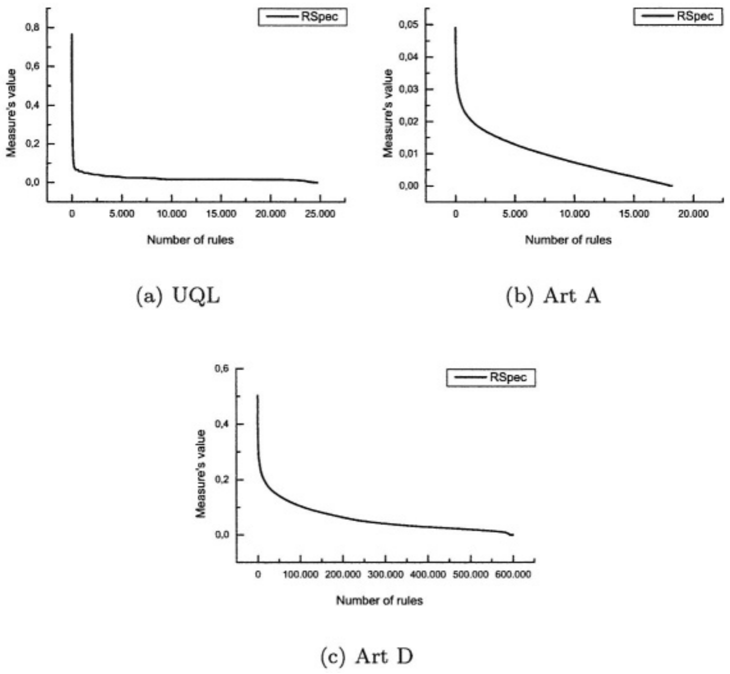


Fig. 3. Behavior of the relative specificity measure

rules. The confidence measure has its maximum value, for more than 50% of the rules, not being suitable for use as an isolated filter.

The combined use of measures increases the restrictive power of the filters. In Table 5 are presented the results for 5% of elevation, the less restrictive percentage and that represents about of 95% of phenomena (measure) in the measure value. It is pointed out that the values in the main diagonal are related to the individual measure use. The other values are relative to the combination of the measures two-by-two. Among the analyzed measures, when combined one with the other, the support presents the most expressive reduction in the number of rules, and the confidence the less expressive one. The smallest number of rules occurred with the combination of the measures relative specificity and support – 1,585 rules selected from the 24,713 rules, representing a reduction of approximately 93%.

Besides the combination of measures two-by-two, the use of the four measures, as a single filter was also explored. The results are described in Table 6. It is observed an expressive reduction in the number of rules in all the situations and, from 10% of elevation (90% of phenomenon's representation), the number of rules is small enough to be presented to a domain's specialist, for an individual rule analysis. It's observed also, that for elevation percentages above 30% the combination of measures is not efficient, since there is practically no reduction on the number of rules.

Table 5. Number of rules filtered for percentage of elevation of 5%

Measure	Confidence	Support	Relative specificity	J_1
Confidence	21,169	2,129	2,751	13,016
Support	2,129	2,137	1,585	1,732
Relative specificity	2,751	1,585	2,888	2,843
J_1	13,016	1,732	2,843	16,139

Table 6. Number of filtered rules, combining five measures

Percentage of reduction	5%	10%	30%	50%	80%
Number of rules	1,355	168	52	11	2

On Table 7 are presented the 11 (eleven) rules obtained applying the 50% percent of elevation (Table 6). These rules are the same of Table 4 for the relative specificity measure.

To evaluate the quality of the filtering process the set of rules were presented to a specialist, the rules from 1 to 4 and from 9 to 10 can be considered as "obvious knowledge" for being representing attributes highly correlated: **dpi_p**, **porpdimp** and **pdp**. The rules 5 to 8 were considered interesting by the specialist, especially rules 7 and 8, which contradict the expectations: for low values of **dpi_p** and **porpdimp**, it was expected values higher of **Pess_Dom**.

5 Final Considerations

The interest about knowledge post-processing has grown as a research object in recent years. This is motivated mainly by the intensification of extracted knowledge uses, in practical applications, and for the great volume of this knowledge that makes impracticable its manual analysis. When extracting patterns with association rules, the considerable amount of extracted knowledge is an even more expressive problem, considering that this technique is known by producing a huge number of rules. The use of objective measures, in the post-processing step to rule evaluation, has as a purpose to establish filters for rule selection, since these measures supply an indication of the hypothetical strength association among *LHS* and *RHS*.

The experiment for validation of measures and the combination of measures as pre-filters or filters for association rule selection was accomplished by, using ten databases with different characteristics and mined with varied parameters, it is possible to consider that the results obtained from this experiment are sufficiently generic to be extrapolated for other cases.

Analyzing the use of measures as a filter – individual measure, combined two-by-two and combination of all – was observed a reduction in the number of rules about 91%, 93% and 95%, when done a elevation of 5% in the measure's

Table 7. Selected rules to be presented to domain expert for evaluation

Rule	LHS → RHS	Conf.	Sup.	Rel. Spec.	J_1
1	dpi_p in (... 12.0%] → porpdimp in (... 3.0%]	1.00	0.76	0.76	0.29
2	porpdimp in (... 3.0%] → dpi_p in (... 12.0%]	1.00	0.76	0.76	0.29
3	dpi_p in (... 12.0%] → porpdp in (99.8 ...]	0.94	0.60	0.46	0.17
4	porpdimp in (... 3.0%] → porpdp in (99.8% ...]	0.93	0.60	0.46	0.17
5	dpi_p in (... 12.0%] → SInstSan in (... 0.5%] and porpdimp in (... 3.0%]	1.0	0.57	0.57	0.22
6	porpdimp in (... 3.0%] → SInstSan in (... 0.5%] and dpi_p in (... 12.0%]	1.0	0.57	0.57	0.22
7	dpi_p in (... 12.0%] → Pess_Dom in (2.5 ... 3.5] and porpdimp in (... 3.0%]	1.00	0.51	0.51	0.20
8	porpdimp in (... 3.0%] → Pess_Dom in (2.5 ... 3.5] and dpi_p in (... 12.0%]	1.00	0.51	0.51	0.20
9	dpi_p in (... 12.0%] → porpdimp in (... 3.0%] and porpdp in (99.8% ...]	1.00	0.60	0.60	0.23
10	porpdimp in (... 3.0%] → dpi_p in (... 12.0%] and porpdp in (99.8% ...]	1.00	0.60	0.60	0.23
11	dpi_p in (... 12.0%] and porpdimp in (... 3.0%] → porpdp in (99.8% ...]	0.93	0.60	0.46	0.17

Legend

dpi_p	– Percentage of improvised permanent domiciles	**SInstSan**	– Percentage of domiciles without sanitary installations
porpdimp	– Percentage of improvised domiciles	**Pess_Dom**	– People average by domiciles
porpdp	– Percentage of permanent domiciles		

minimum value (Tables 4, 5 and 6, respectively). It was also noticed that for the higher percentage of elevation in the measure's minimum value, the reduction in the number of rules was less expressive when combining the measures, becoming equal from 30% of elevation.

In the accomplished work, it is verified that some measures follow the Pareto principle and others do not, and therefore they are more appropriated to the filters constitution. Others measures can be combined and used as "pre-filters" or as complementary measures. In addition, the combined use of measures can extremely reduce the number of rules. Since the rules with high values in one measure don't necessarily has a high value in other measure. Thus, the intersection of rule sets selected for each measure produces a smaller rule set, but it should be adopted with attention to elevated percentages of elevation in the minimum values of the measures.

According to the expert evaluation were identified interesting rules, useful to the set of rules presented, which represent about 30% of the number of rules selected. It's good to remember that the applied filter was highly restrictive, reducing the number of rules from 24.713 to just 11 ones. Filters with less intensity will produce selected sets of a higher size, but still possible to be interpreted by the specialist, and certainly with more interesting rules.

This way, it was verified that the proposed method allows to prioritize the analysis of the rules with higher contribution to the phenomenon (measures), and that they are considered for having higher potential to be interesting. This method acquires more functionality when incorporated into an interactive analysis system, in which the user can vary the parameters, select different measures and combinations of them, and observe if the results are satisfactory.

References

1. Baesens, B., Viaene, S., Vanthienen, J.: Post-processing of association rules. In: Proceedings of the special workshop on post-processing, The Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'2000), Boston, MA, USA (2000) 2–8
2. Liu, B., Hsu, W., Chen, S., Ma, Y.: Analyzing the subjective interestingness of association rules. IEEE Intelligent Systems & their Applications **15** (2000) 47–55
3. Semenova, T., Hegland, M., Graco, W., Williams, G.: Effectiveness of mining association rules for identifying trends in large health databases. In Kurfess, F.J., Hilario, M., eds.: Workshop on Integrating Data Mining and Knowledge Management, The 2001 IEEE International Conference on Data Mining - ICDM'0l, San Jose, California, USA (2001)
4. Jorge, A., Poças, J., Azevedo, P.: A post-processing environment for browsing large sets of association rules. In Simoff, S.J., Noirhomme-Fraiture, M., Böhlen, M.H., eds.: 2ndInternational Workshop on Visual Data Mining, 6th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD-02), Helsinki, Finland (2002) 43–54
5. Bruzzese, D., Davino, C.: Visual post-analysis of association rules. In Simoff, S.J., Noirhomme-Fraiture, M., Böhlen, M.H., eds.: 2ndInternational Workshop on Visual Data Mining, 6th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD-02), Helsinki, Finland (2002) 55–66
6. Hilderman, R.J., Hamilton, H.J.: Applying objective interestingness measures in data mining systems. In Zighed, D., Komorowski, J., eds.: Proceedings of the 4th European Symposium on Principles of Data Mining and Knowledge Discovery, Lyon, France, Springer-Verlag (2000) 432–439 Lecture Notes in Artificial Intelligence.
7. Agrawal, R., Imielinski, T., Swami, A.N.: Mining association rules between sets of items in large databases. In Buneman, P., Jajodia, S., eds.: Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C. (1993) 207–216
8. Adamo, J.M.: Data Mining for Association Rules and Sequential Patterns. Springer-Verlag, New York, NY (2001)
9. Wang, K., Tay, S.H.W., Liu, B.: Interestingness-based interval merger for numeric association rules. In: Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD'98), New York City, New York, USA (1998) 121–128

10. Tan, P., Kumar, V., Srivastava, J.: Selecting the right interestingness measure for association patterns. In: Proceedings of the Eight ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. (2002)
11. **Lavrač**, N., Flach, P., Zupan, R.: Rule evaluation measures: A unifying view. In Dzeroski, S., Flach, P., eds.: Proceedings of the Ninth International Workshop on Inductive Logic Programming (ILP-99). Volume 1634., Springer-Verlag (1999) 74–185 Lecture Notes in Artificial Intelligence.
12. Melanda, E.A., Rezende, S.O.: Investigação do comportmento gráfico de medidas para regras de associação. Relatório Técnico 230, ICMC-USP, São Carlos (2004)
13. Kohavi, R., Brodley, C., Frasca, B., Mason, L., Zheng, Z.: KDD-Cup 2000 organizers' report: Peeling the onion. SIGKDD Explorations **2** (2000) 86–98 http://www.ecn.purdue.edu/KDDCUP.

Two Partitional Methods for Interval-Valued Data Using Mahalanobis Distances

Renata M.C.R. de Souza, Francisco A.T. de Carvalho, and Camilo P. Tenorio

Centro de Informatica - CIn / UFPE,
Av. Prof. Luiz Freire, s/n - Cidade Universitaria,
CEP: 50740-540 - Recife - PE - Brasil
{rmcrs, fatc fcds}@cin.ufpe.br

Abstract. Two dynamic cluster methods for interval data are presented: the first method furnish a partition of the input data and a corresponding prototype (a vector of intervals) for each class by optimizing an adequacy criterion based on Mahalanobis distances between vectors of intervals and the second is an adaptive version of the first method. In order to show the usefulness of these methods, synthetic and real interval data sets considered. The synthetic interval data sets are obtained from quantitative data sets drawn according to bi-variate normal distributions The adaptive method outperforms the non-adaptive one concerning the average behaviour of a cluster quality measure.

1 Introduction

Cluster analysis have been widely used in numerous fields including pattern recognition, data mining and image processing. Their aim is to group data into clusters such that objects within a cluster have high degree of similarity whereas objects belonging to different clusters have high degree of dissimilarity.

The dynamic cluster algorithm [4] is a partitional clustering method whose aim is to obtain both a single partition $P = (C_1, \ldots, C_K)$ of the input data into K clusters and its corresponding set of representatives or prototypes $L = (L_1, \ldots, L_K)$ by locally minimizing an adequacy criterion *W(P, L)* which measures the fitting between the clusters and their representation. This criterion is defined as:

$$W(P, L) = \sum_{k=1}^{K} \Delta_k(L_k) \quad (1)$$

where $\Delta(L_k)$ measure the adequation between the cluster C_k and the its representative L_k.

The k-means algorithm with class prototype updated after all objects have been considered for relocation, is a particular case of dynamic clustering with adequacy function equal to squared error criterion such that class prototypes equal to clusters centers of gravity [7].

In the adaptive version of the dynamic cluster method [3], at each iteration there is a different measure to the comparison of each cluster with its own representation. The advantage of these adaptive distances is that the clustering algorithm is able to recognize clusters of different shapes and sizes. The optimization problem is now to find a partition $P = (C_1, \ldots, C_K)$ of the input data into K clusters, its corresponding set of prototypes $L = (L_1, \ldots, L_K)$ and a set of distances $d = (d_1, \ldots, d_K)$ by locally minimizing an adequacy criterion

$$W(P, L, d) = \sum_{k=1}^{K} \Delta_k(L_k, d_k) \tag{2}$$

where $\Delta(L_k, d_k)$ measure the adequation between the cluster C_k and the its representative L_k using the distance d_k.

Often, objects to be clustered are represented as a vector of quantitative features. However, the recording of interval data has become a common practice in real world applications and nowadays this kind of data is widely used to describe objects. Symbolic Data Analysis (SDA) is a new area related to multivariate analysis and pattern recognition, which has provided suitable data analysis methods for managing objects described as a vector of intervals [1].

Concerning dynamical cluster algorithms for interval data, SDA has provided suitable tools. Verde et al [10] introduced a algorithm considering context dependent proximity functions and Chavent and Lechevalier [2] proposed a algorithm using an adequacy criterion based on Hausdorff distance. Moreover, in [9] is presented an adaptive dynamic cluster algorithm for interval data based on City-block distances.

The aim this paper is to introduce two dynamic cluster methods for interval data. The first method furnishes a partition of the input data and a corresponding prototype (a vector of intervals) for each class by optimizing an adequacy criterion which is based on Mahalanobis distances between vectors of intervals. The second is an adaptive version of the first method.

In these methods, the prototype of each cluster is represented by a vector of intervals, where the bounds of each interval are respectively, for a fixed variable, the average of the set of lower bounds and the average of the set of upper bounds of the intervals of the objects belonging to the cluster for the same variable. In order to show the usefulness of these methods, several synthetic interval data sets ranging from different degree of difficulty to be clustered and an application with a real data set were considered. The evaluation of the clustering results is based on an external validity index.

This paper is organized as follow. In sections 2 and 3 are presented the non-adaptive and adaptive dynamical cluster algorithms for interval data based on Mahalanobis distances, respectively. In section 4 is presented the experimental results with several synthetic interval data sets and an application with a real data set. Finally, in section 5 are given the conclusions.

2 A Dynamical Cluster Method

Let $E = \{s_1, \ldots, s_n\}$ be a set of n patterns described by p interval variables. Each pattern s_i $(i = 1, \ldots, n)$ is represented as a vector of intervals $\mathbf{x}_i = ([a_i^1, b_i^1], \ldots, [a_i^p, b_i^p])^T$. Each cluster C_k $(k = 1, \ldots, K)$ has a prototype L_k that is also represented as a vector of intervals $\mathbf{y}_k = ([\alpha_k^1, \beta_k^1], \ldots, [\alpha_k^p, \beta_k^p])^T$.

Let $\mathbf{x}_{iL} = (a_i^1, \ldots, a_i^p)^T$ and $\mathbf{x}_{iU} = (b_i^1, \ldots, b_i^p)^T$ be two vectors, respectively, of the lower and upper bounds of the intervals describing $\mathbf{x}_i$. Consider also $\mathbf{y}_{kL} = (\alpha_k^1, \ldots, \alpha_k^p)^T$ and $\mathbf{y}_{kU} = (\beta_k^1, \ldots, \beta_k^p)^T$ be two vectors, respectively, of the lower and upper bounds of the intervals describing $\mathbf{y}_k$.

We define the Mahalanobis distance between the two vectors of intervals $\mathbf{x}_i$ and $\mathbf{y}_k$ as:

$$d(\mathbf{x}_i, \mathbf{y}_k) = d(\mathbf{x}_{iL}, \mathbf{y}_{kL}) + d(\mathbf{x}_{iU}, \mathbf{y}_{kU}) \tag{3}$$

where

$$d(\mathbf{x}_{iL}, \mathbf{y}_{kL}) = (\mathbf{x}_{iL} - \mathbf{y}_{kL})^T \mathbf{M}_L (\mathbf{x}_{iL} - \mathbf{y}_{kL}) \tag{4}$$

is the Mahalanobis distance between the two vectors $\mathbf{x}_{iL}$ and $\mathbf{y}_{kL}$ and,

$$d(\mathbf{x}_{iU}, \mathbf{y}_{kU}) = (\mathbf{x}_{iU} - \mathbf{y}_{kU})^T \mathbf{M}_U (\mathbf{x}_{iU} - \mathbf{y}_{kU}) \tag{5}$$

is the Mahalanobis distance between the two vectors $\mathbf{x}_{iU}$ and $\mathbf{y}_{kU}$.

The matrices $\mathbf{M}_L$ and $\mathbf{M}_U$ are defined, respectively, as:

(i) $\mathbf{M}_L = (\det(\mathbf{Q}_{poolL}))^{1/p}\, \mathbf{Q}_{poolL}^{-1}$, where $\mathbf{Q}_{poolL}$ is the pooled covariance matrix with $\det(\mathbf{Q}_{poolL}) \neq 0$, i.e.,

$$\mathbf{Q}_{poolL} = \frac{(n_1 - 1)\mathbf{S}_{1L} + \ldots + (n_K - 1)\mathbf{S}_{KL}}{n_1 + \ldots + n_K - K} \tag{6}$$

In equation (6), $\mathbf{S}_{kL}$ is the covariance matrix of the set of vectors $\{\mathbf{x}_{iL}/i \in C_k\}$ and n_k is the cardinal of C_k $(k = 1, \ldots, K)$.

(ii) $\mathbf{M}_U = (\det(\mathbf{Q}_{poolU}))^{1/p}\, \mathbf{Q}_{poolU}^{-1}$, where $\mathbf{Q}_{poolU}$ is the pooled covariance matrix with $\det(\mathbf{Q}_{poolU}) \neq 0$, i.e.,

$$\mathbf{Q}_{poolU} = \frac{(n_1 - 1)\mathbf{S}_{1U} + \ldots + (n_k - 1)\mathbf{S}_{KU}}{n_1 + \ldots + n_K - K} \tag{7}$$

In equation (7), $\mathbf{S}_{kU}$ is the covariance matrix of the set of vectors $\{\mathbf{x}_{iU}/s_i \in C_k\}$ and n_k is again the cardinal of C_k $(k = 1, \ldots, K)$.

2.1 The Optimization Problem

According to the standard dynamic cluster algorithm, our method look for a partition $P = (C_1, \ldots, C_K)$ of a set of objects into K clusters and its corresponding set of prototypes $L = (L_1, \ldots, L_K)$ by locally minimizing an adequacy criterion usually defined in the following way:

$$W_1(P,L) = \sum_{k=1}^{K} \Delta_k(L_k) = \sum_{k=1}^{K} \sum_{i \in C_k} d(\mathbf{x}_i, \mathbf{y}_k) \tag{8}$$

where $d(\mathbf{x}_i, \mathbf{y}_k)$ is a distance measure between a pattern $s_i \in C_k$ and the class prototype L_k of C_k.

In this method the optimization problem is stated as follows: find the vector of intervals $\mathbf{y}_k = ([\alpha_k^1, \beta_k^1], \ldots, [\alpha_k^p, \beta_k^p])$ which locally minimizes the following adequacy criterion:

$$\Delta_k(L_k) = \sum_{i \in C_k} (\mathbf{x}_{iL} - \mathbf{y}_{kL})^T \mathbf{M}_L (\mathbf{x}_{iL} - \mathbf{y}_{kL}) + \sum_{i \in C_k} (\mathbf{x}_{iU} - \mathbf{y}_{kU})^T \mathbf{M}_U (\mathbf{x}_{iU} - \mathbf{y}_{kU}) \tag{9}$$

The problem now becomes to find the two vectors $\mathbf{y}_{kL}$ and $\mathbf{y}_{kU}$ minimizing the criterion $\Delta_k(L_k)$. According to [5], the solution for $\mathbf{y}_{kL}$ and $\mathbf{y}_{kU}$ are obtained from the Huygens theorem. They are, respectively, the mean vector of the sets $\{\mathbf{x}_{iL}/s_i \in C_k\}$ and $\{\mathbf{x}_{iU}/s_i \in C_k\}$.

Therefore, $\mathbf{y}_k$ is a vector of intervals whose bounds are, for each variable j, respectively, the average of the set of lower bounds and the average of the set of upper bounds of the intervals of the objects belonging to the cluster C_k.

2.2 The Algorithm

The dynamic cluster algorithm with non-adaptive Mahalanobis distance has the following steps:

1. *Initialization.* Randomly choose a partition $\{C_1 \ldots, C_K\}$ of E.
2. *Representation Step.* For $k = 1$ to K compute the vector $\mathbf{y}_k = ([\alpha_k^1, \beta_k^1], \ldots, [\alpha_k^p, \beta_k^p])$ where α_k^j is the average of $\{a_i^j/s_i \in C_k\}$ and β_k^j is the average of $\{b_i^j/s_i \in C_k\}$, $j = 1, \ldots, p$.
3. *Allocation Step.*

```
test ← 0

for $i = 1$ to $n$ do
    define the cluster $C_{k*}$ such that
        $k* = arg\ \min_{k=1,\ldots,K} (\mathbf{x}_{iL} - \mathbf{y}_{kL})^T \mathbf{M}_L (\mathbf{x}_{iL} - \mathbf{y}_{kL}) +$
                                   $(\mathbf{x}_{iU} - \mathbf{y}_{kU})^T \mathbf{M}_U (\mathbf{x}_{iU} - \mathbf{y}_{kU})$
    if $i \in C_k$ and $k* \neq k$
        test ← 1
        $C_{k*} \leftarrow C_{k*} \cup \{s_i\}$
        $C_k \leftarrow C_k \setminus \{s_i\}$
```

4. *Stopping Criterion.*
   If *test* = 0 then STOP, else go to (2).

3 An Adaptive Dynamical Cluster Method

The dynamic cluster algorithm with adaptive distances [3] has also a representation and an allocation step but there is a different distance associated to each cluster. According to the intra-class structure of the cluster C_k, we consider here an adaptive Mahalanobis distance between an object s_i and a prototype L_k, which is defined as:

$$\begin{aligned} d_k(\mathbf{x}_i, \mathbf{y}_k) = & (\mathbf{x}_{iL} - \mathbf{y}_{kL})^T \mathbf{M}_{kL} (\mathbf{x}_{iL} - \mathbf{y}_{kL}) + \\ & (\mathbf{x}_{iU} - \mathbf{y}_{kU})^T \mathbf{M}_{kU} (\mathbf{x}_{iU} - \mathbf{y}_{kU}) \end{aligned} \tag{10}$$

where $\mathbf{M}_{kL}$ and $\mathbf{M}_{kU.}$ are matrices associated to the cluster C_k, both of determinant equal to 1.

3.1 The Optimization Problem

The algorithm looks for a partition in K clusters, its corresponding K prototypes and K different distances associated with the clusters by locally minimizing an adequacy criterion which is usually stated as:

$$W_2(P, L, d) = \sum_{k=1}^{K} \Delta_k(L_k, d_k) = \sum_{k=1}^{K} \sum_{i \in C_k} d_k(\mathbf{x}_i, \mathbf{y}_k) \tag{11}$$

where $d_k(\mathbf{x}_i, \mathbf{y}_k)$ is an adaptive dissimilarity measure between a pattern $s_i \in C_k$ and the class prototype L_k of C_k.

The optimization problem has two stages:

a) The class C_k and the matrices $\mathbf{M}_{kL}$ and $\mathbf{M}_{kU}$ $(k = 1, \ldots, K)$ are fixed. We look for the prototype L_k of the class C_k which locally minimizes

$$\begin{aligned} \Delta_k(L_k, d_k) = & \sum_{i \in C_k} (\mathbf{x}_{iL} - \mathbf{y}_{kL})^T \mathbf{M}_{kL} (\mathbf{x}_{iL} - \mathbf{y}_{kL}) + \\ & \sum_{i \in C_k} (\mathbf{x}_{iU} - \mathbf{y}_{kU})^T \mathbf{M}_{kU} (\mathbf{x}_{iU} - \mathbf{y}_{kU}) \end{aligned} \tag{12}$$

As we know from subsection 2.1, the solutions for α^j_{kL} and β^j_{kU} are, respectively, the average of $\{a^j_i, s_i \in C_k\}$, the lower bounds of the intervals $[a^j_i, b^j_i]$, $s_i \in C_k$, and the average of $\{b^j_i, s_i \in C_k\}$, the upper bounds of the intervals $[a^j_i, b^j_i]$, $s_i \in C_k$.

b) The class C_k and the prototypes L_k $(k = 1, \ldots, K)$ are fixed. We look for the distance d_k of the class C_k which locally minimizes the criterion $\Delta_k(L_k, d_k)$ with $\det(\mathbf{M}_{kL}) = 1$ and $\det(\mathbf{M}_{kU}) = 1$.

According to [3], the solutions are: $\mathbf{M}_{kL} = (\det \mathbf{Q}_{kL})^{1/p}\, \mathbf{Q}_{kL}^{-1}$ where $\mathbf{Q}_{kL}$ is the covariance matrix of the lower bounds of the intervals belonging to the class C_k with $\det(\mathbf{Q}_{kL}) \neq 0$ and $\mathbf{M}_{kU} = (\det \mathbf{Q}_{kU})^{1/p}\, \mathbf{Q}_{kU}^{-1}$ where $\mathbf{Q}_{kU}$ is the covariance matrix of the upper bounds of the intervals belonging to the class C_k with $\det(\mathbf{Q}_{kU}) \neq 0$.

3.2 The Algorithm

The initialization, the allocation step and the stopping criterion are nearly the same in the adaptive and non-adaptive dynamic cluster algorithm. The main difference between these algorithms occurs in the representation step when it is computed for each class k, $(k = 1, \ldots, K)$ the matrices $\mathbf{M}_{kL} = (\det(\mathbf{Q}_{kL}))^{1/p}$ $\mathbf{Q}_{kL}^{-1}$ and $\mathbf{M}_{kU} = (\det(\mathbf{Q}_{kU}))^{1/p}$ $\mathbf{Q}_{kU}^{-1}$.

Remark. If a single number is considered as an interval with equal lower and upper bounds, the results furnished by these symbolic-oriented methods are identical to those furnished by the standard numerical ones [3] when usual data (vector of single quantitative values) are used. Indeed, the clusters and the respective prototypes are identical.

4 Experimental Results

To show the usefulness of these methods, two experiments with synthetic interval data sets of different degrees of clustering difficulty (clusters of different shapes and sizes, linearly non-separable clusters, etc) and an application with a real data set are considered.

The experiments with artificial data sets have three stages: generation of usual and interval data (stages 1 and 2), and evaluation of the clustering results in the framework of a Monte Carlo experience. In each experiment, initially, we considered two standard quantitative data sets in $\Re^2$. Each data set has 450 points scattered among four clusters of unequal sizes and shapes: two clusters with ellipsis shapes and sizes 150 and two clusters with spherical shapes of sizes 50 and 100. Each data point (z_1, z_2) of each one of these artificial quantitative data sets is a seed of a vector of intervals (rectangle): $([z_1 - \gamma_1/2, z_1 + \gamma_1/2], [z_2 - \gamma_2/2, z_2 + \gamma_2/2])$. These parameters γ_1, γ_2 are randomly selected from the same predefined interval. The intervals considered in this paper are: [1, 8], [1,16], [1, 24], [1, 32], and [1, 40].

4.1 Experiment with Synthetic Data Set 1

In this experiment, the data points of each cluster in each quantitative data set were drawn according to a bi-variate normal distribution with correlated components. Data set 1, showing well-separated clusters, is generated according to the following parameters:

a) Class 1: $\mu_1 = 28$, $\mu_2 = 22$, $\sigma_1^2 = 100$, $\sigma_{12} = 21$, $\sigma_2^2 = 9$ and $\rho_{12} = 0.7$;
b) Class 2: $\mu_1 = 65$, $\mu_2 = 30$, $\sigma_1^2 = 9$, $\sigma_{12} = 28.8$, $\sigma_2^2 = 144$ and $\rho_{12} = 0.8$;
c) Class 3: $\mu_1 = 45$, $\mu_2 = 42$, $\sigma_1^2 = 9$, $\sigma_{12} = 6.3$, $\sigma_2^2 = 9$ and $\rho_{12} = 0.7$;
d) Class 4: $\mu_1 = 38$, $\mu_2 = -1$, $\sigma_1^2 = 25$, $\sigma_{12} = 20$, $\sigma_2^2 = 25$ and $\rho_{12} = 0.8$;

Data set 2, showing overlapping clusters, is generated according to the following parameters:

a) Class 1: $\mu_1 = 45$, $\mu_2 = 22$, $\sigma_1^2 = 100$, $\sigma_{12} = 21$, $\sigma_2^2 = 9$ and $\rho_{12} = 0.7$;
b) Class 2: $\mu_1 = 65$, $\mu_2 = 30$, $\sigma_1^2 = 9$, $\sigma_{12} = 28.8$, $\sigma_2^2 = 144$ and $\rho_{12} = 0.8$;
c) Class 3: $\mu_1 = 57$, $\mu_2 = 38$, $\sigma_1^2 = 9$, $\sigma_{12} = 6.3$, $\sigma_2^2 = 9$ and $\rho_{12} = 0.7$;
d) Class 4: $\mu_1 = 42$, $\mu_2 = 12$, $\sigma_1^2 = 25$, $\sigma_{12} = 20$, $\sigma_2^2 = 25$ and $\rho_\rho 12 = 0.8$;

Figures 1 shows, respectively, interval data set 1 with well separated clusters and interval data set 2 with overlapping clusters.

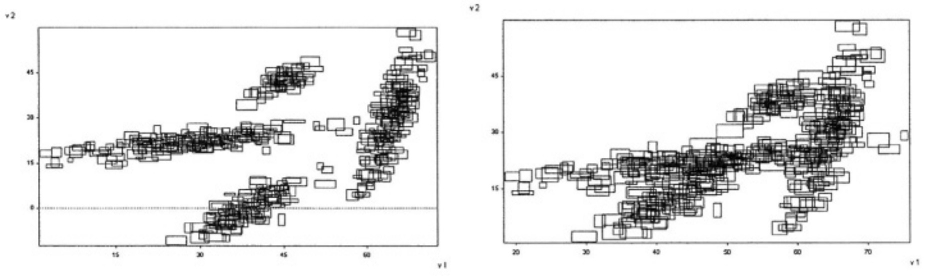


Fig. 1. Symbolic data showing well-separated classes and overlapping classes

4.2 Experiment with Synthetic Data Set 2

Here, the data points of each cluster in each quantitative data set were also drawn according to a bi-variate normal distribution but its components are non-correlated. Data set 3, showing well-separated clusters, is generated according to the following parameters:

a) Class 1: $\mu_1 = 28$, $\mu_2 = 22$, $\sigma_1^2 = 100$, $\sigma_{12} = 0$, $\sigma_2^2 = 9$ and $\rho_{12} = 0.0$;
b) Class 2: $\mu_1 = 67$, $\mu_2 = 30$, $\sigma_1^2 = 9$, $\sigma_{12} = 0$, $\sigma_2^2 = 144$ and $\rho_{12} = 0.0$;
c) Class 3: $\mu_1 = 45$, $\mu_2 = 45$, $\sigma_1^2 = 9$, $\sigma_{12} = 0$, $\sigma_2^2 = 9$ and $\rho_{12} = 0.0$;
d) Class 4: $\mu_1 = 38$, $\mu_2 = -7$, $\sigma_1^2 = 25$, $\sigma_{12} = 0$, $\sigma_2^2 = 25$ and $\rho_{12} = 0.0$;

Data set 4, showing overlapping clusters, is generated according to the following parameters:

a) Class 1: $\mu_1 = 45$, $\mu_2 = 22$, $\sigma_1^2 = 100$, $\sigma_{12} = 0$, $\sigma_2^2 = 9$ and $\rho_{12} = 0.0$;
b) Class 2: $\mu_1 = 60$, $\mu_2 = 30$, $\sigma_1^2 = 9$, $\sigma_{12} = 0$, $\sigma_2^2 = 144$ and $\rho_{12} = 0.0$;
c) Class 3: $\mu_1 = 52$, $\mu_2 = 38$, $\sigma_1^2 = 9$, $\sigma_{12} = 0$, $\sigma_2^2 = 9$ and $\rho_{12} = 0.0$;
d) Class 4: $\mu_1 = 42$, $\mu_2 = 12$, $\sigma_1^2 = 25$, $\sigma_{12} = 0$, $\sigma_2^2 = 25$ and $\rho_\rho 12 = 0.0$;

Figures 2 shows, respectively, interval data set 3 with well separated clusters and interval data set 4 with overlapping clusters.

4.3 The Monte Carlo Experience

The evaluation of these clustering methods was performed in the framework of a Monte Carlo experience: 100 replications are considered for each interval data set,

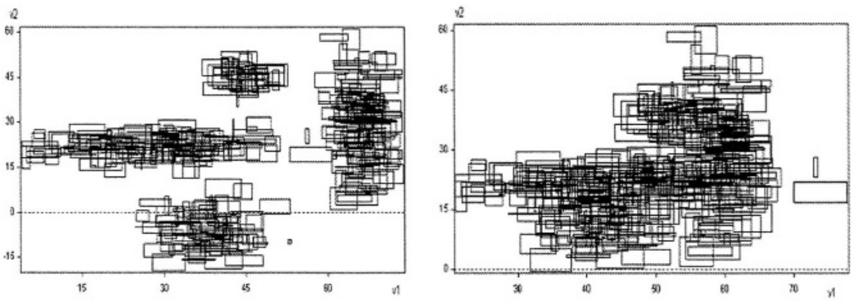


Fig. 2. Symbolic data showing well-separated classes and overlapping classes

as well as for each predefined interval. In each replication a clustering method is run (until the convergence to a stationary value of the adequacy criterion W_1 or W_2) 50 times and the best result, according to the criterion W_1 or W_2, is selected.

Remark. As in the standard Mahalanobis (adaptive and non-adaptive) distance methods for dynamic cluster, these methods have sometimes a problem with the inversion of matrices. When this occurs, the actual version of these algorithms stops the current iteration and re-starts a new one. The stopped iteration is not take into account among the 50 which should be run.

The average of the corrected Rand (CR) index [6] among these 100 replications is calculated. The CR index assesses the degree of agreement (similarity) between an a priori partition (in our case, the partition defined by the seed points) and a partition furnished by the clustering algorithm. CR can take values in the interval [-1,1], where the value 1 indicates a perfect agreement between the partitions, whereas values near 0 (or negatives) correspond to cluster agreements found by chance [8].

Table 1 and 2 show the values of the average CR index according to the different methods and interval data sets. These tables also show suitable (null and alternative) hypothesis and the observed values of statistics following a Student's t distribution with 99 degrees of freedom.

Table 1. Comparison between the clustering methods with interval data sets 1 and 2

Range of values of γ_i $i = 1, 2$	Interval Data Set 1			Interval Data Set 2		
	Non-Adaptive Method	Adaptive Method	$H_0 : \mu_1 \leq \mu$ $H_a : \mu_1 > \mu$	Non-Adaptive Method	Adaptive Method	$H_0 : \mu_1 \leq \mu$ $H_a : \mu_1 > \mu$
$\gamma_i \in [1, 8]$	0.778	0.996	80.742	0.409	0.755	13.266
$\gamma_i \in [1, 16]$	0.784	0.986	82.182	0.358	0.688	22.609
$\gamma_i \in [1, 24]$	0.789	0.963	61.464	0.352	0.572	20.488
$\gamma_i \in [1, 32]$	0.802	0.937	39.181	0.349	0.435	18.204
$\gamma_i \in [1, 40]$	0.805	0.923	29.084	0.341	0.386	9.2851

Table 2. Comparison between the clustering methods with interval data sets 3 and 4

Range of values of γ_i $i = 1,2$	Interval Data Set 3			Interval Data Set 4		
	Non-Adaptive Method	Adaptive Method	$H_0 : \mu_1 \leq \mu$ $H_a : \mu_1 > \mu$	Non-Adaptive Method	Adaptive Method	$H_0 : \mu_1 \leq \mu$ $H_a : \mu_1 > \mu$
$\gamma_i \in [1,8]$	0.779	0.995	70.618	0.365	0.530	22.360
$\gamma_i \in [1,16]$	0.789	0.995	57.295	0.378	0.497	18.207
$\gamma_i \in [1,24]$	0.792	0.995	52.325	0.388	0.474	18.289
$\gamma_i \in [1,32]$	0.793	0.988	43.664	0.398	0.449	8.799
$\gamma_i \in [1,40]$	0.806	0.974	28.914	0.373	0.389	2.177

As the interval data set used to calculate the CR index by each method in each replication is exactly the same, the comparison between the proposed clustering methods is achieved by the paired Student's t-test at a significance level of 5%. In these tests, μ_1 and μ are, respectively, the average of the CR index for adaptive and non-adaptive methods.

From the results in Tables 1 and 2, it can be seen that the average CR indices for the adaptive method are greater than those for the non-adaptive method in all situations. In addition, the statistic tests support the hypothesis that the average performance (measured by the CR index) of the adaptive method is superior to the non-adaptive method.

4.4 Application with Real Data

A data set with 33 car models described by 8 interval variables is used in this application. These car models are grouped in two a priori clusters of unequal sizes: one cluster (Utilitarian or Berlina) of size 18 and another cluster (Sporting or Luxury) of size 13. The interval variables are: Price, Engine Capacity, Top Speed, Acceleration, Step, Length, Width and Height.

The CR indices obtained from the clustering results are, respectively, 0.242 and 0.126 for adaptive and non-adaptive methods. From these results, we can conclude that, for this data set, the adaptive method surpass the non-adaptive method concerning this clustering quality measure.

5 Conclusions

In this paper, dynamic cluster methods for interval data are presented. Two methods are considered: the first method furnish a partition of the input data and a corresponding prototype for each class by optimizing an adequacy criterion which is based on Mahalanobis distances between vectors of intervals. The second is an adaptive version of the first method. In both methods the prototype of each class is represented by a vector of intervals, where the bounds of these intervals for a variable are, respectively, the average of the set of lower bounds and the average of the set of upper bounds of the intervals of the objects belonging to the class for the same variable.

The convergence of these algorithms and the decrease of their partitioning criterions at each iteration is due to the optimization of their adequacy criterions. The accuracy of the results furnished by these clustering methods were assessed by the corrected Rand index considering synthetic interval data sets in the framework of a Monte Carlo experience and an application with a real data set. Concerning the average CR index for synthetic interval data sets, the method with adaptive distance clearly outperforms the method with non-adaptive distance. This was also the case for the car data set.

Acknowledgments. The authors would like to thank CNPq (Brazilian Agency) for its financial support.

References

1. Bock, H.H. and Diday3, E.: Analysis of Symbolic Data: Exploratory Methods for Extracting Statistical Information from Complex Data. Springer, Berlin, (2000)
2. Chavent, M. and Lechevallier, Y.: Dynamical Clustering Algorithm of Interval Data: Optimization of an Adequacy Criterion Based on Hausdorff Distance. In: Sokolowsky and H.H. Bock Eds., K. Jaguja, A. (eds) Classification, Clustering and Data Analysis (IFCS2002). Springer, Berlin, (2002) 53–59
3. Diday, E. and Govaert, G.: Classification Automatique avec Distances Adaptatives. R.A.I.R.O. Informatique Computer Science, **11** (4) (1977) 329–349
4. Diday, E. and Simon, J.C.: Clustering analysis. In: K.S. Fu (ed) Digital Pattern Clasification. Springer, Berlin et al, (1976) 47–94
5. Govaert, G.: Classification automatique et distances adaptatives. Thèse de 3ème cycle, Mathématique appliquée, Université Paris VI (1975)
6. Hubert, L. and Arabie, P.: Comparing Partitions. Journal of Classification, **2** (1985) 193–218
7. Jain, A.K., Murty, M.N. and Flynn, P.J. 1999. Data Clustering: A Review. ACM Computing Surveys, 31, (3), 264-323
8. Milligan, G. W.:Clustering Validation: results and implications for applied analysis In: Arabie, P., Hubert, L. J. and De Soete, G. (eds) Clustering and Classification, Word Scientific, Singapore, (1996) 341–375
9. Souza, R.M.C.R. and De Carvalho, F. A. T.: Clustering of interval data based on city-block distances. Pattern Recognition Letters, **25** (3) (2004) 353–365
10. Verde, R., De Carvalho, F.A.T. and Lechevallier, Y.: A dynamical clustering algorithm for symbolic data. In: Diday, E., Lechevallier, Y. (eds) Tutorial on Symbolic Data Analysis (Gfkl2001), (2001) 59–72

A Classifier for Quantitative Feature Values Based on a Region Oriented Symbolic Approach

Simith T. D'Oliveira Junior, Francisco A. T. de Carvalho, and Renata M. C. R. de Souza

Centro de Informatica - CIn / UFPE, Av. Prof. Luiz Freire, s/n - Cidade Universitaria, CEP: 50740-540 - Recife - PE - Brasil
{stdj, fatc, rmcrs}@cin.ufpe.br

Abstract. In this paper, a classifier for quantitative feature values (intervals and/or points) based on a region oriented symbolic approach is proposed. In the learning step, each class is described by a region (or a set of regions) in $\Re^p$ defined by a convex hull. To affect a new object to a class a dissimilarity matching function compares the class description (a region or a set of regions) with a point in $\Re^p$. Experiments with two artificial data sets generated according to bi-variate normal distributions have been performed in order to show the usefulness of this classifier. The evaluation of the proposed classifier is accomplished according to the calculation of the prediction accuracy (error rate), speed and storage measurements computed through of a Monte Carlo simulation method with 100 replications.

1 Introduction

Supervised classification aims to construct classification rules from examples, with known class structures, which allow to assign new objects to classes [4]. With the explosive growth in the use of databases new classification approaches have been proposed. *Symbolic Data Analysis* (SDA) [1] has been introduced as a new domain related to multivariate analysis, pattern recognition and artificial intelligence in order to extend classical exploratory data analysis and statistical methods to symbolic data. Symbolic data allows multiple (sometimes weighted) values for each variable, and it is why new variable types (interval, categorical multi-valued and modal variables) have been introduced. These new variables allow to take into account variability and/or uncertainty present in the data.

Ichino et al. [5] and Yaguchi et al. [8] introduced a symbolic classifier as a region oriented approach for quantitative, categorical, interval and categorical multi-valued data. This approach is an adaptation of the concept of *mutual neighbours* introduced in [3] to define the concepts of *mutual neighbours* between symbolic data and *Mutual Neighbourhood Graph* between groups. At the end of the learning step, the symbolic description of each group is obtained through the use of an approximation of a *Mutual Neighbourhood Graph (MNG)* and a symbolic join operator. In the allocation step, an observation is assigned to a particular group based on a dissimilarity matching function.

Souza et al. [7] and De Carvalho et al. [2] have proposed another *MNG* approximation to reduce the complexity of the learning step without losing the classifier performance in terms of prediction accuracy. Concerning the allocation step, a classification rule, based on new similarity and dissimilarity measures, have been introduced to carry out the assignment of an individual to a class.

In this paper, we introduce a new symbolic classifier for symbolic data based on a region oriented approach. Here, we are concerned with symbolic data that are represented by quantitative feature vectors. More general symbolic data type can be found in [1]. Let $\Omega = \{\omega_1, \cdots, \omega_n\}$, be a set of n individuals described by p quantitative features $X_j (j = 1, \ldots, p)$. Each individual ω_i $(i = 1, \ldots, n)$ is represented by a quantitative feature vector $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip})$, where x_{ij} is a *quantitative feature value.* A quantitative feature value may be either a continuous value (e.g., $x_{ij} = 1.80$ meters of height) or an interval value (e.g., $x_{ij} = [0, 2]$ hours, the duration of a student evaluation).

Table 1 shows an example of a data table, where each individual is represented by two continuous feature values x_1 and x_2 and a categorical feature that represents its class.

At end of the learning step, the description of each class is a region (or a set of regions) in $\Re^p$ defined by the convex hull of the objects belonging to this class which is obtained through a suitable approximation of a *Mutual Neighbourhood Graph (MNG),* This approach aims to reduce the over-generalization that is produced when each class is described by a region (or a set of regions) defined by the hyper-cube formed by the objects belonging to this class and then to improve the accuracy performance of the classifier. The assignment of a new object to a class is based on a dissimilarity matching function which compares the class description with a point in $\Re^p$.

Section 2 presents the concepts of regions and graphs. In sections 3 and 4, the learning and allocation steps of this symbolic classifier are presented, respectively. In order to show the usefulness of this approach, two artificial data sets generated according to bi-variate normal distributions are classified. To evaluate the performance of the new classifier, prediction accuracy, speed and storage measurements are computed in the framework a Monte Carlo experience. Section 5 describes the experiments and the performance analysis and section 6 gives the concluding remarks.

Table 1. A data table for $n = 6$ individuals and $p = 2$ continuous values

Individuals	Continuous Feature Values		Class
	x_1	x_2	
ω_1	0.189566	-0.080602	0
ω_2	0.203821	-0.390586	0
ω_3	0.448117	2.672457	1
ω_4	0.477598	3.188989	1
ω_5	4.102095	2.757729	2
ω_6	2.355855	3.749226	2

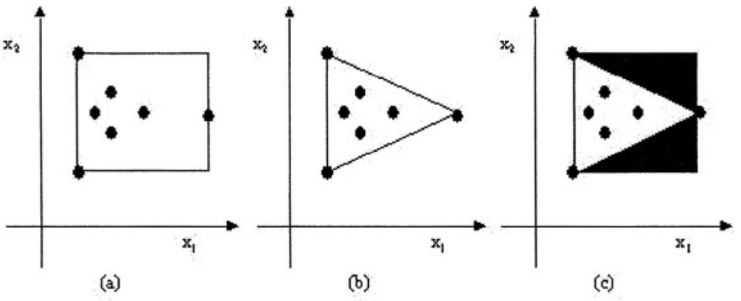


Fig. 1. (a) *J-region,* (b) *H-region,* (c) over-generalization

2 Concepts of Regions and Graphs

Let $C_k = \{\omega_{k1}, \ldots, \omega_{kN_k}\}, k = 1, \ldots, m$, be a class of individuals with $C_k \cap C_{k'} = \emptyset$ if $k \neq k'$ and $\cup_{k=1}^{m} C_k = \Omega$. The individual $\omega_{kl}, l = 1, \ldots,\ N_k$, is represented by the continuous feature vector $\mathbf{x}_{kl} = (x_{kl1}, \ldots, x_{klp})$.

A symbolic description of the class C_k can be obtained by using the join operator (Ichino et al (1996)).

Definition 1. The join between the continuous feature vectors $\mathbf{x}_{kl}$ $(l = 1, \ldots, N_k)$ is an interval feature vector which is defined as $\mathbf{y}_k = \mathbf{x}_{k1} \oplus \ldots \oplus \mathbf{x}_{kN_k} = (x_{k11} \oplus \ldots \oplus x_{kN_k1}, \ldots, x_{k1j} \oplus \ldots \oplus x_{kN_kj}, \ldots, x_{k1p} \oplus \ldots \oplus x_{kN_kp})$, where $x_{k1j} \oplus \ldots \oplus x_{kN_kj} = [min\{x_{k1j}, \ldots, x_{kN_kj}\}, max\{x_{k1j}, \ldots, x_{kN_kj}\}](j = 1, \ldots, p)$.

Moreover, we can associate to each class C_k two regions in $\Re^p$: one spanned by the join of its elements and another spanned by the convex hull of its elements.

Definition 2. The *J-region* associated to class C_k is a region in $\Re^p$ which is spanned by the join of the objets belonging to class C_k and it is defined as $R_J(C_k) = \{\mathbf{x} \in \Re^p : min\{x_{k1j}, \ldots, x_{kN_kj}\} \leq x_j \leq max\{x_{k1j}, \ldots, x_{kN_kj}\}, j = 1, \ldots, p\}$. The volume associated to the hyper-cube defined by the region $R_J(C_k)$ is $\pi(R_J(C_k))$.

Definition 3. The *H-region* associated to class C_k is a region in $\Re^p$ which is spanned by the convex hull formed by the objects belonging to class C_k and it is defined as $R_H(C_k) = \{\mathbf{x} = (x_1, \ldots, x_j, \ldots, x_p) \in \Re^p : \mathbf{x}$ is inside the envelop of the convex hull defined by the continuous feature vectors $\mathbf{x}_{kl} = (x_{kl1}, \ldots, x_{klp}), l = 1, \ldots N_k\}$. The volume associated to the internal points inside the convex hull envelop defined by $R_H(C_k)$ is $\pi(R_H(C_k))$.

Figure 1 illustrates the description of a class by a *J-region* and by a *H-region.* It is clear that the representation based on a *J-region* (see Ichino et al (1996), Souza et al (1999) and De Carvalho et al (2000)) over-generalizes the class description given by a *H-region.* It is why in this paper this last option will be used.

The *mutual neighbourhood graph (MNG)* (Ichino et al (1996)) yields information about interclass structure.

Definition 4. The objects belonging to class C_k are each one *mutual neighbours* (Ichino et al (1996)) if $\forall \omega_{k'l} \in C_{k'}$ $(k' \in \{1,\ldots,m\}, k' \neq k), \mathbf{x}_{k'l} \notin R_J(C_k)$ $(l = 1,\ldots,N_{k'})$. In this case, the *MNG* of C_k against $\overline{C_k} = \cup_{k'=1, k'\neq k}^{m} C_{k'}$, which is constructed by joining all pairs of objects which are mutual neighbours, is a complete graph.

If the objects belonging to class C_k are not each one mutual neighbours, we look for all the subsets of C_k where its elements are each one mutual neighbours and which are a *maximal clique* in the *MNG,* which, in that case, is not a complete graph. To each of these subsets of C_k we can associate a *J-region* and calculate the volume of the corresponding hyper-cube defined by it.

In this paper we introduce another definition to the *MNG.*

Definition 5. The objects belonging to class C_k are each one *mutual neighbours* if $\forall \omega_{k'l} \in C_{k'}, k' \in \{1,\ldots,m\}, k' \neq k, \mathbf{x}_{k'l} \notin R_H(C_k)(l = 1,\ldots,N_{k'})$. The *MNG* of C_k against $\overline{C_k} = \cup_{k'=1, k'\neq k}^{m} C_{k'}$, defined in this way is also a complete graph.

If the objects belonging to class C_k are not each one mutual neighbours, again we look for all the subsets of C_k where its elements are each one mutual neighbours and which are a maximal clique in the *MNG* and to each of these subsets of C_k we can associate a *H-region* and calculate the volume of the corresponding convex-hull defined by it.

3 Learning Step

This step aims to provide a description of each class through a region (or a set of regions) in $\Re^p$ defined by the convex hull formed by the objects belonging to this class, which is obtained through a suitable approximation of a Mutual Neighbourhood Graph *(MNG).* Concerning this step, we have two basic remarks:

a) The first is that a difficulty arises when the *MNG* of a class C_k is not complete. In this case, we look for all the subsets of C_k where its elements form a *maximal clique* in the *MNG.* However, it is well known that the computational complexity in time to find all maximal cliques on a graph is exponential and the construction of an approximation of the *MNG* is necessary.
b) The second concerns the kind of region that is suitable to describe a class C_k.

3.1 The Learning Algorithm

The construction of the *MNG* for the classes C_k $(k = 1,\ldots,m)$ and the representation of each class by a *H-region* (or by a set of *H-region*) is accomplished in the following way:

```
For $k = 1,\ldots,m$ do
  1 Find the region $R_H(C_k)$ (according to definition 3) associated to class
    $C_k$ and verify if the objects belonging to this class are each one mutual
    neighbors according to the definition 5
  2 If it is the case, construct the MNG (which is a complete graph) and stop
```

```
3 If it is not the case (MNG approximation) do:
    3.1 choose an object of $C_k$ as a seed according to the lexicographic order of these objects in $C_k$; do $t = 1$ and put the seed in the set $C_k^t$; remove the seed from $C_k$
    3.2 add the next object of $C_k$ (according to the lexicographic order) to $C_k^t$ if all the objects belonging now to $C_k^t$ remains each one mutual neighbors according to the definition 5; if this is true, remove this object from $C_k$
    3.3 repeat step 3.2) for all remaining objects in $C_k$
    3.4 Find the region $R_H(C_k^t)$ (according to definition 3) associated to $C_k^t$)
    3.5 if $C_k \neq \emptyset$, do $t = t + 1$ and repeat steps 3.1) to 3.4) until $C_k = \emptyset$
4 construct the MNG (which now is not a complete graph) and stop
```

At the end of this algorithm it is computed the subsets $C_k^1, \ldots, C_k^{n_k}$ of class C_k and it is obtained the description of this class by the *H-regions* $R_H(C_k^1), \ldots, R_H(C_k^{n_k})$. Figure 2 shows an example for the case of two classes.

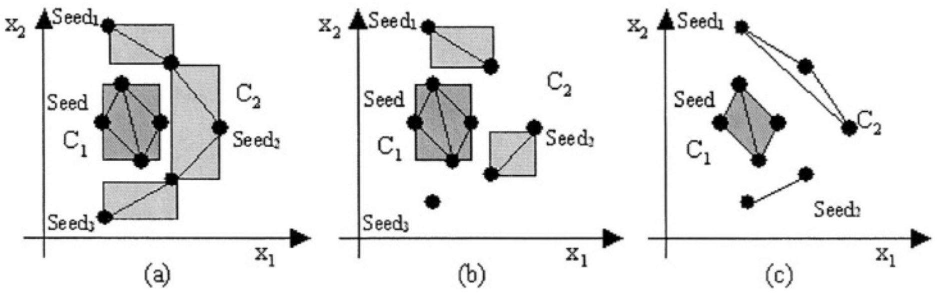


Fig. 2. (a) MNG approximation and *J-sets* (Ichino et al (1996)), (b) MNG approximation and *J-sets* (Souza et al (1999), De Carvalho et al (2000)), (c) MNG approximation and *H-sets* (proposed approach)

4 Allocation Step

The aim of the allocation step is to associate a new object to a class based on a dissimilarity matching function that compares the class description (a region or a set of regions) with a point in $\Re^p$.

Let ω be a new object, which is candidate to be assigned to a class $C_k (k = 1, \ldots, m)$, and its corresponding description given by the continuous feature vector $\mathbf{x} = (x_1, \ldots, x_p)$. Remember that from the learning step it is computed the subsets $C_k^1, \ldots, C_k^{n_k}$ of C_k.

The *classification rule* is defined as follow: ω is affected to the class C_k if

$$\delta(\omega, C_k) \leq \delta(\omega, C_h), \forall h \in \{1, \ldots, m\} \tag{1}$$

where $\delta(\omega, C_h) = min\{\delta(\omega, C_h^1), \ldots, \delta(\omega, C_h^{n_h})\}$.

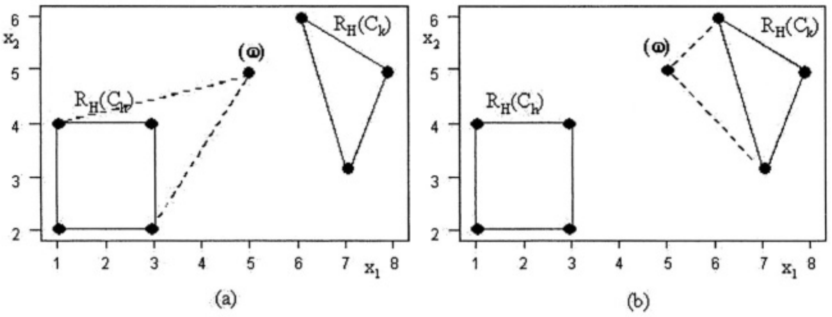


Fig. 3. Differences in area

In this paper, the dissimilarity matching function δ is defined as

$$\delta(\omega, C_h^s) = \frac{\pi(R_H(C_h^s \cup \{\omega\}) - \pi(R_H(C_h^s)}{\pi(R_H(C_h^s \cup \{\omega\})}, \ s = 1, \ldots, n_h \tag{2}$$

Example: Consider two classes: $C_k = \{\omega_1, \omega_2, \omega_3, \omega_4\}$ where where $\mathbf{x}_1 = (1,2)$, $\mathbf{x}_2 = (3,2), \mathbf{x}_3 = (1,4), \mathbf{x}_4 = (3,4)$ and $C_h = \{\omega_5, \omega_6, \omega_7\}$ where $\mathbf{x}_5 = (6,6), \mathbf{x}_6 = (7,3), \mathbf{x}_7 = (8,5)$. Let ω be a new individual where $\mathbf{x}_\omega = (5,5)$. Using the function given by the equation (2), we obtain $\delta(\omega, C_k) = 0.43$ and $\delta(\omega, C_h) = 0.44$. According to the classification rule, the new individual ω is affected to class C_k. Figures 3(a) and 3(b) show the differences in area if the individual ω is associated to the C_k $(C_k \cup \{\omega\})$ or the C_h $(C_h \cup \{\omega\})$.

5 Experiments and Performance Analysis

Experiments with two artificial quantitative data sets in $\Re^2$ and a corresponding performance analysis of the classifier introduced in this paper are considered in this section. In these experiments, 100 replications of each set with identical statistical properties are obtained for each one of the training and test sets. The data points of each cluster in each data set were drawn according to a bi-variate normal distribution with correlated components.

5.1 Experiment 1

In this experiment, the data set has 300 points scattered among three clusters of equal sizes and unequal shapes: two clusters with ellipsis shapes and sizes 100 and one cluster with spherical shapes of size 100.

Data set 1 (Fig. 4) with three clusters is generated according to the following parameters:

a) Class 1: $\mu_1 = 0$, $\mu_2 = 0$, $\sigma_1^2 = 4$, $\sigma_{12} = 1.7$, $\sigma_2^2 = 1$ and $\rho_{12} = 0.85$;
b) Class 2: $\mu_1 = 0$, $\mu_2 = 3$, $\sigma_1^2 = 0.25$, $\sigma_{12} = 0.0$, $\sigma_2^2 = 0.25$ and $\rho_{12} = 0.0$;
c) Class 3: $\mu_1 = 4$, $\mu_2 = 3$, $\sigma_1^2 = 4$, $\sigma_{12} = -1.7$, $\sigma_2^2 = 1$ and $\rho_{12} = -0.85$;

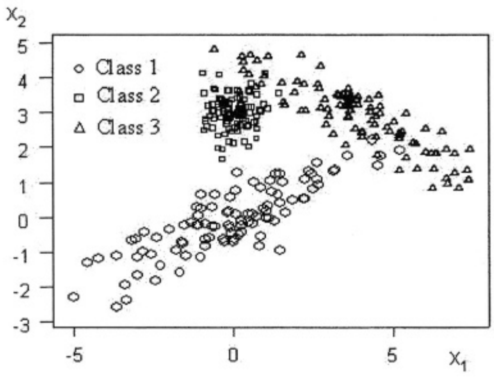


Fig. 4. Quantitative data with three classes

5.2 Experiment 2

Here, the data set has 500 points scattered among five clusters of equal sizes and unequal shapes: three clusters with ellipsis shapes and sizes 100 and two clusters with spherical shapes of sizes 100.

Data set 2 (Fig. 5) with five clusters is generated according to the following parameters:

a) Class 1: $\mu_1 = 0$, $\mu_2 = 0$, $\sigma_1^2 = 4$, $\sigma_{12} = 1.7$, $\sigma_2^2 = 1$ and $\rho_{12} = 0.85$;
b) Class 2: $\mu_1 = 0$, $\mu_2 = 3$, $\sigma_1^2 = 0.25$, $\sigma_{12} = 0.0$, $\sigma_2^2 = 0.25$ and $\rho_{12} = 0.0$;
c) Class 3: $\mu_1 = 4$, $\mu_2 = 3$, $\sigma_1^2 = 4$, $\sigma_{12} = -1.7$, $\sigma_2^2 = 1$ and $\rho_{12} = -0.85$;
c) Class 4: $\mu_1 = 8.5$, $\mu_2 = 5.5$, $\sigma_1^2 = 4$, $\sigma_{12} = 1.7$, $\sigma_2^2 = 1$ and $\rho_{12} = 0.85$;
b) Class 5: $\mu_1 = 7.5$, $\mu_2 = 3.0$, $\sigma_1^2 = 0.25$, $\sigma_{12} = 0.0$, $\sigma_2^2 = 0.25$ and $\rho_{12} = 0.0$;

5.3 Performance Analysis

The evaluation of our method, called here *H-region approach,* where class representation, MNG approximation and dissimilarity matching function are based on *H-regions,* is performed based on prediction accuracy, speed and storage in comparison with the approach where class representation, MNG approximation and dissimilarity matching function are based on *J-regions* (called here *J-region approach*).

The prediction accuracy of the classifier was measured through the error rate of classification obtained from a test set. The estimated error rate of classification corresponds to the average of the error rates found between the 100 replications. Speed and storage were also assessed, respectively, from the 100 replications, by the average time in minutes spent in the learning and allocation steps and the average memory in k-bytes spent to store the regions.

To compare these methods, we use paired Student's t-tests at the significance level of 5%. Tables 2, 3 and 4 show, respectively, comparisons between these

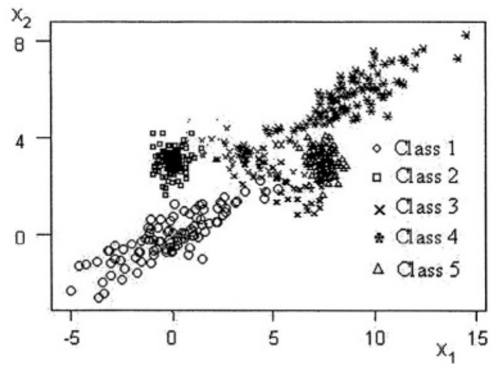


Fig. 5. Quantitative data with five classes

Table 2. Comparison between the classifiers according to the average error rate

Quantitative data sets	*H-region approach*		*J-region approach*		Hypothesis	Decision
	average error	standard deviation	average error	standard deviation	$H_0 : \mu_1 \geq \mu_2$ $H_1 : \mu_1 < \mu_2$	
three classes	2.57	0.991	3.16	1.062	-5.5642	Reject H_0
five classes	4.716	1.086	5.29	1.134	-5.171	Reject H_0

classifiers according to the average error rate, the average time and the average memory for the data sets with three and five classes. In all these tables, the statistics test follow a Student's t distribution with 99 degrees of freedom and μ_1 and μ_2 are, respectively, the corresponding averages for the *H-region* approach and the *J-region* approach.

From the values in Table 2, we can conclude that for these data sets the average error rate for *H-region* approach is lower than that for the *J-region* approach. Also, from the test statistics we can see that the *H-region* approach outperforms the *J-region* approach on prediction accuracy. From Tables 3 and 4, we can see that the average time and average memory values calculated using *H-region approach* are greater than those using the *J-region approach* in both data sets and paired Student's t-tests at the significance level of 5% support the hypothesis that the *J-region approach* outperforms the *H-region approach.*

The complexity in time of *H-region approach* is $O(n^2 logn)$ (see [6]), whereas the complexity of the *J-region approach* is $O(n^2)$ (see[2]), n being the cardinality of the learning set. In conclusion, concerning the speed and storage, the *J-region approach* clearly outperforms *H-region approach* as the size of data set grows. However, *H-region approach* presents better results concerning prediction accuracy.

Table 3. Comparison between the classifiers according to the average time

Quantitative data sets	*H-region approach*		*J-region approach*		Hypothesis	Decision
	average error	standard deviation	average error	standard deviation	$H_0 : \mu_1 \geq \mu_2$ $H_1 : \mu_1 < \mu_2$	
three classes	23.93	3.66	7.29	0.47	44.4236	Accept H_0
five classes	63.06	7.46	12.33	0.47	68.088	Accept H_0

Table 4. Comparison between the classifiers according to the average memory

Quantitative data sets	*H-region approach*		*J-region approach*		Hypothesis	Decision
	average error	standard deviation	average error	standard deviation	$H_0 : \mu_1 \geq \mu_2$ $H_1 : \mu_1 < \mu_2$	
three classes	44110	271.8	27441	184	817.993	Accept H_0
five classes	73859	405.5	46037	255.8	975.198	Accept H_0

6 Concluding Remarks

A classifier for quantitative feature values based on a region oriented symbolic approach is presented in this paper. The input of this classifier is a set of continuous feature vectors. Concerning the learning step, each class is described by a region (or a set of regions) in $\Re^p$ defined by the convex hull formed by the objects belonging to this class, which is obtained through a suitable approximation of a Mutual Neighbourhood Graph *(MNG)*. The goal of this step is to reduce the over-generalization that is produced when each class is described by a region (or a set of regions) in $\Re^p$ defined by the hyper-cube formed by the objects belonging to this class and in this way to improve the classifier's prediction accuracy. In the allocation step, to assign a new object to a class, a dissimilarity matching function, which compares a class description (a region or a set of regions) with an individual description (point in $\Re^p$), was introduced.

To show the usefulness of this approach, two artificial quantitative data sets in $\Re^2$ are considered presenting three and five classes. These data sets were drawn according to bi-variate normal distributions with correlated components. The evaluation of the proposed classifier was based on prediction accuracy, and speed and storage measurements in comparison with the classifier which uses in the learning and allocation steps the *J-region*. The prediction accuracy was evaluated according to the error rate of classification. These measurements were calculated in the framework of a Monte Carlo experience with 100 replications and were obtained from independent test sets. The results showed that, concerning the speed and storage measurements, the *J-region approach* outperforms *H-region approach*. However, *H-region approach* furnishes better results concerning prediction accuracy.

Acknowledgments. The authors would like to thank CNPq (Brazilian Agency) for its financial support.

References

1. Bock, H.H. and Diday, E.: Analysis of Symbolic Data: Exploratory Methods for Extracting Statistical Information from Complex Data. Springer, Berlin Heidelberg (2000)
2. De Carvalho, F.A.T., Anselmo, C.A.F. and Souza, R.M.C.R.: Symbolic approach to classify large data sets, In: Kiers, H.A.L. et al (Eds.) Data Analysis, Classification, and Related Methods, , Springer, (2000) 375–380
3. Gowda, K. C and Krishna, G.: Agglomerative clustering using the concept of mutual nearest neighbourhood. Pattern Recognition, 10, 105–112
4. Hand, D. J. Construction and Assessment of Classification Rules. Willey (1997)
5. Ichino, M., Yaguchi, H. and Diday, E.: A fuzzy symbolic pattern classifier In: Diday, E. et al (Eds.): Ordinal and Symbolic Data Analysis. Springer, Berlin, (1996) 92–102
6. O'Rourke, J.: Computational Geometry in C (Second Edition), Cambridge University Press (1998)
7. Souza, R. M. C. R., De Carvalho, F. A. T. and Frery, A. C.: Symbolic approach to SAR image classification. IEEE 1999 International Geoscience and Remote Sensing Symposium, Hamburgo, (1999) 1318–1320
8. Yaguchi, Ichino, M., H. and Diday, E.: A knowledge accquisition system based on the Cartesian space model. In: Diday, E. et al (Eds.): Ordinal and Symbolic Data Analysis. Springer, Berlin, (1996) 113–122

The Protein Folding Problem Solved by a Fuzzy Inference System Extracted from an Artificial Neural Network

Eduardo Battistella and Adelmo Luis Cechin

Universidade do Vale do Rio dos Sinos - Unisinos,
PIPCA - Programa Interdisciplinar de Computação Aplicada,
Av. Unisinos 950 - 93022-000 - São Leopoldo, RS - Brasil - Caixa-Postal: 270
{eduardob, cechin}@exatas.unisinos.br

Abstract. This paper reports the results of a rule extraction process from an Artificial Neural Network (ANN) used to predict the backbone dihedral angles of proteins based on physical-chemical attributes. By analyzing the fuzzy inference system extracted from the knowledge acquired by the ANN we want to scientifically explain part of the results obtained by the scientific community when processing the Hydropathy Index and the Isoeletric Point and also show that the rule extraction process from ANNs is an important tool that should be more frequently used. To obtain these results we defined a methodology that allowed us to formulate hypothesis statistically sustained and to conclude that these attributes are not enough to predict the backbone dihedral angles when processed by an ANN approach.

1 Introduction

Since Anfinsen [1] first stated that the information necessary for protein folding resides completely within its primary structure an intensive research has been done to corroborate this theory. Thirty years later, one of the major goals of computational biology still is to understand the relationship between the amino acid sequence and the structure of a protein. If this relationship were known, then the structure of a protein could be reliably predicted. Unfortunately this relationship is not that simple.

These last three decades had exposed a gradual growth in the complexity of computational techniques engaged in the problem of protein folding. Among various empirical methods, based on databases where structures and sequences are known, the use of Artificial Neural Networks (ANNs) demonstrated evidences of being a suitable technique provided that it learns from a set of examples over which shows a capacity to generalize the acquired knowledge to unseen data. This characteristic makes the ANN a fitted technique for processing data from domains where there are little or incomplete knowledge about the problem to be solved but there are enough data for designing a model.

This search led to variations in the way that the amino acid primary sequence has been coded. As compiled by Wu & McLarty [2], most of the works use a variety of orthogonal codifications to represent each one of the 20 amino acids. Other works, in minor scale, use the amino acid physical-chemical attributes to represent them. Unfortunately, the use of these attributes did not bring improvements in the accuracy. Cherkauer & Shavlik [3] observe "... *our most surprising (and disappointing) discovery is that the ability to choose among thousands of features does not appear to lead to significant gains on the secondary structure task"*. Baldi & Brunak [4] affirm *"If one wants to use real-numbered quantification of residue hydrophobicity, volume, charge, and so on, one should be aware of the harmful impact it can have on the input space"*.

In this context, this paper reports the results obtained by extracting rules from an ANN used to predict the backbone dihedral angles of proteins (*phi* and *psi* angles) based on physical-chemical attributes. By analyzing the fuzzy inference system extracted from the ANN, when processing the Hydropathy Index (HP scale) and the Isoeletric Point (pI), we want to scientifically explain part of the "disappointing" results reported by Cherkauer & Shavlik [3] and the "harmful impact" cited by Baldi & Brunak [4]. To obtain these results we defined and followed a methodology composed by 5 main steps exposed over the next sections: database definition; network training; rule extraction; results interpretation; and hypothesis formulation.

This paper is structured as follows. In section 2 we present the database definition of which a list of proteins is built and the attributes are defined. The training process of the ANN is presented in section 3 and the rule extraction process is introduced in section 4. Section 5 presents the interpretation of the results and the next section closes the methodology with the hypothesis formulation. Finally, in section 7 we present the conclusions.

2 Database Definition

Concerning the database used in the training process, we adopted a validated one. The EVA project [5] is a web-based server that performs a weekly automatic evaluation of the "*n*" newest experimental structures added to Protein Data Bank (PDB)[1]. One of its various outputs is an up to date list of non-homologous domains. At the time the EVA database was accessed, 2.980 non-homologous domains were identified. We adopted the EVA database due to the possibility of scaling from a fewer to a higher number of examples as needed.

Once the database is identified we need to define the attributes to be processed by the ANN and over which the knowledge will be extracted. Among various physical-chemical attributes available in the literature we opted by the Hydropathy Index (HP scale) and the Isoeletric Point (pI). The choice of these attributes was biased by their importance in the protein folding process reported by most of the classical Biochemistry books (e.g. Lehninger [6], Voet & Voet [7]).

[1] http://www.rcsb.org/pdb

In Table 1 we can view the amino acids and the original values for HP and pI attributes. These values were standardized (i.e. modified to have a mean of 0 and a standard deviation of 1) when submitted to the ANN.

The HP scale, introduced by Kyte & Doolittle [8], is a degree of hydrophobicity of amino acid side chains. It can be used to measure the tendency of an amino acid to seek or avoid an aqueous environment. Lower values represent hydrophilic environment and higher values hydrophobic ones. The pI is based on a relationship between the net electric charge of the amino acid and the pH of the solution. Additional explanations can be found at Lehninger [6], Voet & Voet [7] and others.

Table 1. The amino acids and their values for Hydropathy Index (HP) and Isoeletric Point(pI)

Letter	Abreviation	Name	HP	pI
A	ALA	Alanine	1,8	6.01
C	CYS	Cysteine	2,5	5,07
D	ASP	Aspartate	-3,5	2,77
E	GLU	Glutamate	-3,5	3,22
F	PHE	Phenylalanine	2,8	5,48
G	GLY	Glycine	-0,4	5,97
H	HIS	Histidine	-3,2	7,59
I	ILE	Isoleucine	4,5	6,02
K	LYS	Lysine	-3,9	9,74
L	LEU	Leucine	3,8	5,98
M	MET	Methionine	1,9	5,74
N	ASN	Asparagine	-3,5	5,41
P	PRO	Proline	-1,6	6,48
Q	GLN	Glutamine	-3,5	5,65
R	ARG	Arginine	-4,5	10,76
S	SER	Serine	-0,8	5,68
T	THR	Threonine	-0,7	5,87
V	VAL	Valine	4,2	5,97
W	TRP	Tryptophan	-0,9	5,89
Y	TYR	Tyrosine	-1,3	5,66

3 Training the ANN

A diagram of the basic network is shown in Figure 1. The processing units are arranged in layers, with the input units shown at the top and the output unit on the bottom. The units in the input layer are fully connected with the units on the hidden layer and with the output unit (i.e. short-cut). The units on the hidden layer are fully connected with the unit on the output layer.

To process the residues belonging to the protein's primary structure the windowing technique was used. It consists of a sliding window that is shifted over

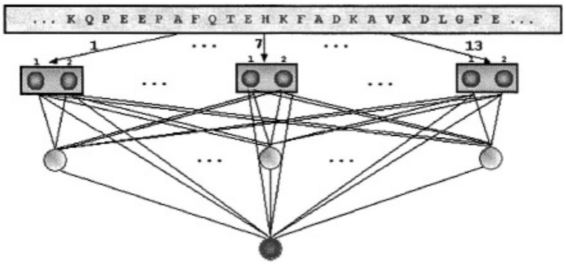


Fig. 1. Architecture of the ANN used

the full sequence from left to right. Each time the window is shifted a new input is generated. The *phi* and *psi* angles of the central residue on the window are the target of prediction.

With the windowing technique we explicitly want to capture local interactions of the amino acids with the aim to explain the behavior of the physical-chemical attributes in short range intervals. Albeit, we have in mind that some influences to the protein folding process are also caused by long distant interactions.

This work implements a network for a function approximation problem instead of a common classification one. It tries to predict the torsion angle *(phi* or *psi)* instead of the traditional *alpha, beta* and *coil* classes. The reason for this approach is to offer a suitable result for algorithms that need a starting point more precise than the information returned by a class. The learning algorithm used in the supervised learning was the Resilient Propagation (RProp).

After some experiments at which were tested the number of hidden units (i.e. 0,1,2,4,8,16), the number of residues covered by the window (i.e. 3,5,7,... ,19) and the seed for random initialization of weights, the final network was defined as: window size of 13 residues and 4 units on the hidden layer. The window size of 13 residues leads to an input layer of 26 units (13 residues * 2 attributes). We noticed a behavior similar to the one reported by Qian & Sejnowski [9] when varying the window size and the number of hidden units. The Root Mean Square (RMS) errors obtained in the training process were: 0.540593 for *phi* and 0.817681 for *psi.*

4 Rule Extraction

The rule extraction algorithm used in this work was the Fuzzy Automatically Generated Neural Inferred System (FAGNIS) [10]. It allows the rule extraction directly from the trained ANN, guarantees the functional equivalence between the ANN and the fuzzy inference system extracted and shows a high comprehensibility of the extracted rules (less rules and simple rules).

The main idea behind FAGNIS can be described as the act of partitioning the neural space into linear regions. It is done like a linear regression by parts as the network is being trained. Each linear region found corresponds to a

Takagi-Sugeno rule [11]. Three outputs are reported to each linear region (rule) generated by FAGNIS: a prototype that characterizes the behavior of the patterns belonging to the rule; a linear equation based on the analysis of the neural space that gives the output of a pattern belonging to the rule; and a membership function that computes the membership value of the pattern.

Takagi-Sugeno rules [11] are based on the fuzzy partitioning of the input space. For each subspace, a linear equivalence between input and output is accomplished. The fuzzy inferred system with input variables $x_1, \ldots, x_n$ and the output variable y, whose value is inferred, contains a rule system in the form:

$$R\text{: } If\ f\,(x_1\ is\ F_1, \ldots, x_n\ is\ F_n)\ \ Then\ y = g\,(x_1 \ldots, x_n) \tag{1}$$

where: $F_1, \ldots, F_n$ are fuzzy sets with linear membership functions representing a fuzzy subspace in which the implication R can be applied for reasoning; f is the logical function that connects the propositions in the premise; and g is the function that implies the value of y when $x_1, \ldots, x_n$ satisfies the premises. The linear consequent has a higher degree of comprehensibility than a nonlinear one (and knowledge acquisition is an important point in this work).

In Figure 2 we can observe how FAGNIS handles a *sigmoid* activation unit. The unit is represented by 3 rules:

$$\left.\begin{array}{lll} If\ x\ is\ F_1 & Then & y = -1 \\ If\ x\ is\ F_2 & Then & y = 0.5x \\ If\ x\ is\ F_3 & Then & y = +1 \end{array}\right\} \tag{2}$$

that show an equivalence with the original *sigmoid* function. Explanations about how to obtain the membership functions can be found at [10].

Another feature available in FAGNIS *is* the statistical approach used to validate the rules extracted. There is no need for a validation set during the learning process of the ANN because FAGNIS uses parameter estimation and confidence intervals. The measure used to compute the solution found is based on the number of parameters effectively used by the ANN: the number of linear regions and the number of parameters of each linear region.

Complementing the information available for each rule we have: a linear equation generated from a linear regression of all patterns belonging to the rule; the

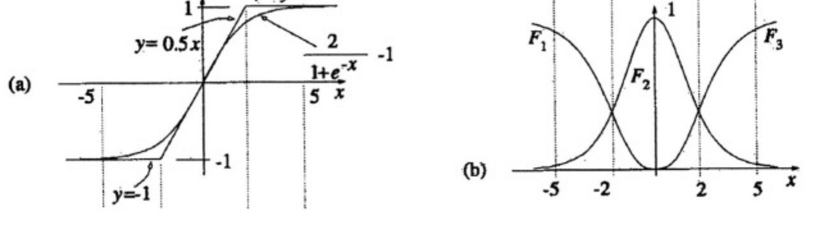


Fig. 2. Sigmoid function and the consequent part of each rule (a); implemented number of membership functions for a given activation unit (b)

percentage of patterns covered by the rule; and the RMS error of the linear equation that is used as a measure of how good is the found solution.

5 Interpreting the Results

After training the networks until the error was stabilized, we extracted 30 rules for *psi* angle and 21 rules for *phi.* For each rule are reported the prototypes that describe the behavior of the residues (concerning HP and pI) in the input window. The prototypes are the main knowledge over which the interpretations and hypothesis will be done.

Two approaches can be used to interpret the rules extracted. First, we can analyze them by the criterion of rules that aggregate the highest number of patterns. Second, rules with the lowest RMS error. Both approaches will be covered here. Due to constraints in the article size we will expose the main idea related to the rule interpretation and not an exhaustive enumeration of all rules.

In the first approach the first 4 rules for *psi* angle include almost 50% (48.83%) of the database. A linear regression for all patterns reported a RMS error of 0.866781. None of the rules extracted has a higher RMS error than the one reported by this "global" linear regression. It means that each rule has a better prediction capacity than the solution obtained with a linear regression over all patterns.

We can observe in Figure 3 the prototype that characterizes the patterns covered by rule number 5. All patterns (i.e. windows of 13 residues) that show equivalent values for HP and pI will be handled by this rule. Concerning the influence of HP over the residues, can be observed a period of 2 hydrophobic residues (higher values) followed by 2 hydrophilic (lower ones). Another point to be observed is the HP of the central residue. It has a high hydrophobic value. Concerning pI, can be observed the neutral characteristic of the central residue. As should be expected the influence of HP and pI becomes weaker in the extremes of the window.

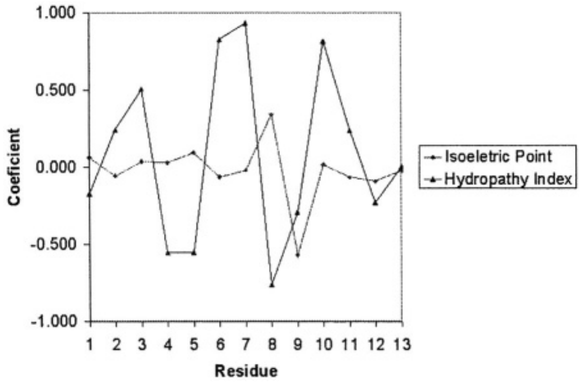


Fig. 3. Prototype of rule 5 where can be observed the influence of Hydropathy Index and Isoeletric Point over the patterns covered by its rule

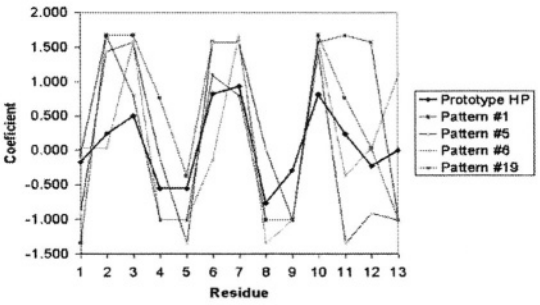


Fig. 4. Hydropathy Index from 4 patterns with a high membership value regarding to rule 5 - plotted to corroborate the information given by the prototype

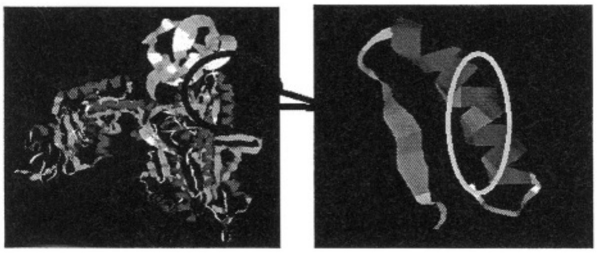


Fig. 5. Threonine tRNA from *Escherichia Coli* and the corresponding 13 residues (within the circle) with a cycle reported by the prototype that characterizes the rule

Corroborating the knowledge given by the prototype we can plot the patterns that have the highest membership value for this rule. This information is obtained during the process of rule extraction. In Figure 4 we can observe the HP of 4 patterns with a high membership value regarding to the rule 5. We can observe that the patterns show a correlated behavior to the prototype.

If we took the pattern #1, reported in Figure 3, and isolate the corresponding window of 13 residues in the protein, we got an *alpha* that belongs to Threonine tRNA from Escherichia Coli (code 1QF6 at PDB). In Figure 5 we can see the full protein structure and the isolated *alpha* at which the hydrophobic residues were accentuated in green. Besides the *alpha* portion we also exposed part of a *beta* strand to demonstrate the alignment of the hydrophobic residues of these two structures. The cycle reported by the prototype, concerning to the hydrophobicity, can be observed in the *alpha*.

The second approach is to analyze the rules extracted with the lowest RMS error. The first rule gives a RMS error considerably low (i.e. 0.246627) but covers just 0.19% of the patterns. This behavior can also be noted for *phi* angle. The explanation to this fact resides in a specialization in the neural space that was perceived by the rule extraction algorithm and transformed in a rule. Specialized regions in the neural space tend to cover fewer patterns.

In the very same way we could extend these graphs, tables and explanations to other rules, to the pI attribute and to *phi* angle. Specifically to the pI, we can affirm that an interesting behavior was observed in some rules on which positive and negative values appear symmetrically over the prototype.

6 Hypothesis Formulation

The patterns used to corroborate the prototype in Figure 3 were not randomly chosen. They represent very important points to understand the RMS error obtained by the rules extracted and to formulate a hypothesis about the knowledge acquired.

Even so the top 172 patterns from rule 5 have a membership value higher than 90% the ones from 1 to 20 per se carry a relevant knowledge. In Table 2 we can analyze patterns number 1, 5, 6, 19 and 20 and observe an alternation between angles that lead to *alpha* and *beta* regions for patterns with very similar and high membership value (i.e. the degree of membership of the pattern to the rule). All patterns from 2 to 4 and from 7 to 18 have angles that lead to *alpha* region.

If we extend this analysis to all patterns from rule 5 we will see what is plotted in Figure 6. Observing the classic Ramachandran plot we can see that the patterns covered by rule 5 are sparse over typical *alpha* and *beta* regions. This plot was generated using the expected *phi* and *psi* torsion angles. It means that rule 5 clusters in the input space patterns that lead to distinct regions in the output space.

Table 2. Top 20 patterns covered by rule 5 ordered by membership value

Patterns #	Membership	phi	psi
1	0.997093	-52.09	-51.39
5	0.994286	-127.88	161.59
6	0.993221	-73.57	-23.06
19	0.981746	-118.25	115.68
20	0.981539	-64.71	-40.96

The knowledge embedded in Table 2 shows that both alpha and beta patterns have similar values for HP and pI considering the input processed window. We can infer that only these attributes are not enough to determine *phi* and *psi* torsion angles (or the secondary structure).

To corroborate this hypothesis we analyzed the patterns processed by the ANN. We were able to find a knowledge already reported by Qian & Sejnowski [9]: the Isoleucine amino acid tends to form a beta structure (with its corresponding *phi* and *psi* angles) while Leucine tends to form alpha. When using a traditional orthogonal representation this behavior does not introduces any problem, since each amino acid has a unique representation. When using a physical-chemical attribute this behavior leads us to a problem during the process of partitioning the input space. Due to the fact that Isoleucine and Leucine

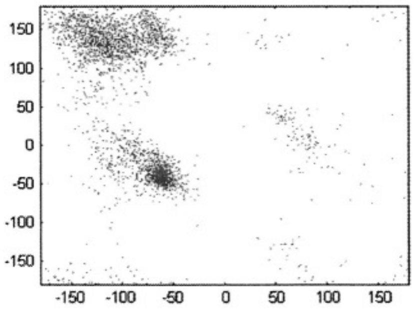


Fig. 6. Patterns covered by rule 5 plotted using the torsion angles - X axis represents *phi* and Y axis represents *psi*

have very close physical-chemical attributes values, the learning process of the ANN captured this information and considered both as similar entries. The rule extraction process captured this information and exposed it when reported the patterns belonging to each rule. This knowledge also explains the RMS error obtained.

7 Conclusion

In this paper we are able to explain why the use of the attributes Hydropathy Index and Isoeletric Point processed by a Neural Network approach are not enough to predict the backbone dihedral angles (*phi* and *psi* angles). To obtain these results we defined and followed a methodology composed by 5 main steps: database definition; network training; rule extraction; results interpretation; and hypothesis formulation.

We exposed the rule extraction process as a viable tool to interpret the knowledge acquired by ANNs. We can guarantee that the rule extraction process is statistically valid but the experimental proof of the biochemical knowledge discovered is beyond the scope of this work and should be done by qualified technicians.

As already stated, we choose HP and pI due to a constant reference in the literature about their importance in the protein folding process. The same methodology defined here could be used to any other physical-chemical attribute.

In future works we plan to analyze different attributes and different input codifications like the common orthogonal one.

Acknowledgments

This work was partially developed in collaboration with HP Brazil R&D. Research supported in part by the FAPERGS, process number 03/50314.8.

References

1. Christian B. Anfinsen. Principles that govern the folding of protein chains. *Science,* 181:223–230, 1973.
2. Cathy H. Wu and Jerry W. McLarty. *Methods in Computational Biology and Biochemestry: Neural Networks and Genome Informatics.* Elsevier, 2000.
3. K. J. Cherkauer and J. W. Shavlik. Protein structure prediction: Selecting salient features from large candidate pools. In AAAI Press, editor, *Proceedings of the First International Conference on Intelligent Systems for Molecular Biology,* pages 74–82, 1993.
4. P. Baldi and S. Brunak. *Bioinformatics: The Machine Learning Approach.* The MIT Press, Cambridge, Massachusetts and London, England, 2001.
5. Burkhard Rost and Volker Eyrich. Eva: Large-scale analysis of secondary structure prediction. *Prot. Struct. Funct. Genet.,* 5:192–199, 2001.
6. Albert L. Lehninger. *Principles of Biochemistry.* Worth, New York, 3 edition, 2000.
7. Donald Voet, Judith G. Voet, and Charlotte W. Pratt. *Biochemistry.* Willey Text Books, 3 edition, 2003.
8. Jack Kyte and Russel F. Doolittle. A simple method for displaying the hydropathic character of a protein. *J. Molec. Biol.* 157:105–132, 1982.
9. N. Qian and T.J. Sejnowski. Predicting the secondary structure of globular proteins using neural networks models. *J. Molec. Biol.,* 202:865–884, 1988.
10. Adelmo Luis Cechin. *The Extraction of Fuzzy Rules from Neural Networks.* PhD thesis, Tbingen, 1998.
11. T. Takagi and M. Sugeno. Fuzzy identification on systems and its application to modeling and control. *IEEE Transactions on Systems, Man, and Cybernetics,* 15:116–132, 1985.

A Multi-robot Strategy for Rapidly Searching a Polygonal Environment

Alejandro Sarmiento[1], Rafael Murrieta-Cid[2], and Seth Hutchinson[1]

[1] University of Illinois at Urbana-Champaign,
Urbana, Illinois, USA
{asarmien, seth}@uiuc.edu
[2] ITESM Campus Estado de México,
Atizapán de Zaragoza, Estado de México, México
rmurriet@itesm.mx

Abstract. In this paper we address the problem of finding an object in a polygonal environment as quickly as possible on average, with a team of mobile robots that can sense the environment.

We show that for this problem, a trajectory that minimizes the distance traveled may not minimize the expected value of the time to find the object. We prove the problem to be NP-hard by reduction, therefore, we propose the heuristic of a utility function. We use this utility function to drive a greedy algorithm in a reduced search space that is able to explore several steps ahead without incurring too high a computational cost. We have implemented this algorithm and present simulation results for a multi-robot scheme.

1 Introduction

The problem of determining a good strategy to accomplish a visibility-based task such as environment modeling [4], pursuit-evasion [7] [8], or object finding [6] [15], is a very challenging an interesting research area. Specially when the sensors are not static but rather are carried by mobile robots.

In this paper we address the problem of finding an object in an unknown location somewhere inside a polygonal environment as quickly as possible on average. For this, we use a team of mobile robots that can sense the environment. This is the optimization problem of minimizing the expected value of time required to find the object, where time is a random variable defined by a search path together with the probability density function associated to the object's location. The possible applications have a wide range, from finding a specific piece of art in a museum to search and rescue of injured people inside a building.

We present a discrete formulation, in which we use a visibility-based decomposition of the polygon to convert the problem into a combinatoric one. We define particular locations from where the robot senses the environment (guards). The guards' visibility regions are used to calculate the probability of seeing an object for the first time from a particular location. These are chosen from an appropriate set which is computed automatically. With this, we are able to abstract the problem to one of finding a path in a graph whose nodes represent the sensing locations. Trajectories are then constructed from arcs in a reduced visibility graph.

We show that for this problem, a trajectory that minimizes the distance traveled may not minimize the expected value of the time to find the object. We prove the problem to be NP-hard by reduction, therefore, we propose the heuristic of a utility function, defined as the ratio of a gain over a cost. We use this utility function to drive a greedy algorithm in a reduced search space that is able to explore several steps ahead without incurring too high a computational cost. We have implemented this algorithm and present simulation results for a multi-robot scheme.

2 Problem Definition

In general terms, we define the problem of searching for an object as follows: Given one or more mobile robots with sensing capabilities, a completely known environment and an object somewhere in the world, develop a motion strategy for the robots to find the object in the least amount of time on average.

The environment W is known, and modeled as a polygon that may contain holes (obstacles). The obstacles generate *both motion and visibility constraints.*

Furthermore, we assume that the probability of the object being in any specific point is uniformly distributed throughout the polygon's interior. Therefore, the probability of the object being in any subset $R \subseteq W$ is proportional to the area of R.

We also assume that we start with a set of locations L (also known as *guards,* from the art gallery problem [9]) so that each point in W can be seen from at least one location in L. There are several criteria for determining the goodness of this set. For example, the minimal number of locations (art gallery problem), locations along the shortest path that covers the whole environment (shortest watchman path [14]), and so on. In [13] we propose an algorithm for determining this set. The basic idea is to place guards inside the region bounded by inflection rays of the aspect graph. These regions have the property that a point inside can see both sides of reflex vertices (those with internal angle greater than π).

The visibility region of location L_j, denoted $V(L_j)$, is the set of points in W that have a clear line of sight to L_j (the line segment connecting them does not intersect the exterior of W). The set L is chosen so that the associated visibility regions define a cover of W. This means that their union is to the whole environment, that is, $\bigcup_j V(L_j) = W$. We do not require nor assume the set L to be minimal.

For the sake of simplicity, *we will first present the basic algorithm for the case of a single robot and then we will extend it for the general multi-robot case.*

Our exploration protocol is as follows: the robot always starts at a particular location in L (the starting point) and visits the other locations as time progresses (it follows the shortest paths between them). It only gathers information about the environment (sensing) when it reaches one of these locations – it does not sense while moving. We describe the route followed by the robot as a series of locations L_{i_k} that starts with the robot's initial location and includes every other location at least once. Note that while L_j refers to locations in the environment, L_{i_k} refers to the *order* in which those locations are visited. That is, the robot always starts at L_{i_0}, and the k-th location it visits is referred to as L_{i_k}.

For any route *R*, we define the time to find the object *T* as the time it takes to go through the locations – in order – until the object is first seen.

Our goal is to find the route that minimizes the expected value of the time it takes to find the object

$$E\left[T|R\right] = \sum_j t_j P\left(T = t_j\right) \tag{1}$$

where

$$P\left(T = t_j\right) = \frac{Area\left(V\left(L_{i_j}\right) \setminus \bigcup_{k<j} V\left(L_{i_k}\right)\right)}{Area(W)}. \tag{2}$$

Here, t_j is the time it takes the robot to go from its initial position – through all locations along the route – until it reaches the j-th *visited* location L_{i_j}, and $P\left(T = t_j\right)$ is the probability of seeing the object for the first time from location L_{i_j}. Since the robot only senses at specific locations, we also denote this probability of seeing the object for the first time from location L_{i_j} as $P\left(L_{i_j}\right)$.

Explicitly, the probability of seeing the object for the first time from a given location is proportional to the visibility polygon of that region $V\left(L_{i_j}\right)$ minus the already explored space up to that point $\bigcup_{k<j} V\left(L_{i_k}\right)$.

2.1 Expected Value Vs. Worst Case

It is important to note the difference between minimizing the expected value of the time to find an object and minimizing the time it would take in the worst case. To minimize the worst case time, the robot must find the shortest path that completely covers the environment (the Shortest Watchman Tour problem [1]). This usually means that no portions of the environment are given any priority over others and the rate at which new portions of the environment are seen is not important.

On the other hand, to minimize the expected value of the time, the robot must gain probability mass of seeing the object as quickly as possible. For a uniform object PDF, this translates into sensing large portions of the environment as soon as possible, even if this means spending more time later to complete covering the whole environment. For non-uniform PDFs, the robot should visit the most promising areas first. We believe this represents another paradigm for coverage tasks, where it is important to gain as much new information in the shortest time as possible. This could be very useful in applications where the time assigned to the task is limited or not completely known.

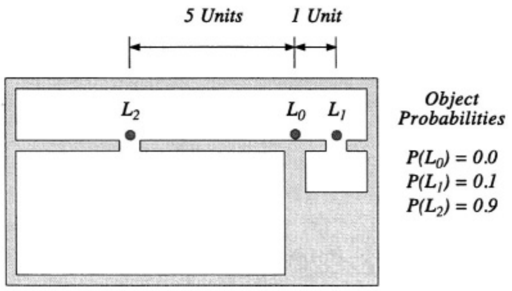


Fig. 1. Example with a simple environment

The trajectories that satisfy the previous two criteria are not the same. In fact, for a given environment, the route that minimizes the distance traveled may not minimize the expected value of the time to find an object along it. Consider the example in Fig. 1.

The robot starts in the corridor at location L_0. The object will always be in one of two rooms, and the probability of it being in either is related to the size of the room.

If the robot goes to the smaller room first and then moves on to the larger room (route 1), it reaches L_1 at time 1 and L_2 at time 7. The expected value of the time it takes to find the object following this route is $E[T|(L_0, L_1, L_2)] = (0.1)(1) + (0.9)(7) = 6.4$. The robot always completes its search after 7 seconds.

On the other hand, if the robot moves to the larger room first and then goes to the smaller room (route 2), it reaches L_2 at time 5 and L_1 at time 11. The expected time in this case is $E[T|(L_0, L_2, L_1)] = (0.9)(5) + (0.1)(11) = 5.6$. In the worst case, it will take the robot 11 seconds to find the object.

A robot following route 1 always finishes searching after 7 seconds, while a robot following route 2 takes 11 seconds. Route 1 minimizes the distance traveled. However, the average time it takes for a robot following route 1 to find the object is 6.4 seconds whereas for route 2 it is only 5.6 seconds. Route 2 minimizes the expected value of the time to find an object.

Thus, *a trajectory that is optimal in the distance traveled does not necessarily minimize the expected value of the time to find an object along it.*

3 Proposed Solution

Since we assume that we are given a set of sensing locations that completely cover the environment, we are interested in finding an order of visiting those locations – the problem becomes a combinatorial search.

In general, the robot will not be able to travel between two locations by following a straight line. In this cases, we use a reduced visibility graph [10] and Dijkstra's Algorithm to follow the shortest path between them.

3.1 Reduction from an NP-Hard Problem

The *Minimum Weight Hamiltonian Path Problem,* known to be NP-hard [3], can be reduced to the problem of finding the optimal visiting order of sensing locations which minimizes the expected time to find an object.

In order to make a formal reduction, we abstract the concept of environment and visibility regions. We only consider a set of locations that have an associated probability of seeing the object and whose visibility regions do not overlap.

The reduction consists in defining the distance between the sensing locations as the edge weights of the Minimum Weight Hamiltonian Path Problem and setting the probabilities uniformly (same value for all).

Since the probabilities are set uniformly, the route that minimizes the expected time will be exactly the same as the one that minimizes the distance traveled. This happens because the expected value of the time to find an object is determined only by the time it takes to reach locations along the route. Since time is proportional to distance, the route that minimizes time will also minimize the distance.

Given that the solutions to both problems are the same ordering of locations, finding a polynomial algorithm to solve these instances of the defined problem would also solve the Minimum Weight Hamiltonian Path Problem in polynomial time, thereby proving that the proposed problem is NP-hard.

3.2 Utility Heuristic

Since trying to find an optimal solution is a futile effort, we have decided to implement an iterative, greedy strategy, one that tries to achieve a good result by exploring just a few steps ahead.

We propose a greedy algorithm, called *utility greedy,* that tries to maximize a utility function. This function measures how convenient it is to visit a determined location from another, and is defined as follows:

$$U\left(L_j, L_k\right) = \frac{P\left(L_k\right)}{Time\left(L_j, L_k\right)}. \tag{3}$$

This means that if the robot is currently in L_j, the utility of going to location L_k is proportional to the probability of finding the object there and inversely proportional to the time it must invest in traveling.

A robot using this function to determine its next destination will tend to prefer locations that are close and/or locations where the probability of seeing the object is high. Intuitively, it is convenient to follow such a strategy, but its relationship with the expected value minimization will be more evident after the following analysis.

Consider a definition of expectation for a non-negative random variable, such as time, from [11]

$$E[T|R] = \int_0^\infty P(T > t)dt = \int_0^\infty (1 - P(T \leq t))\, dt = \int_0^\infty (1 - F_T)\, dt \tag{4}$$

in which F_T is a *cumulative distribution function.*

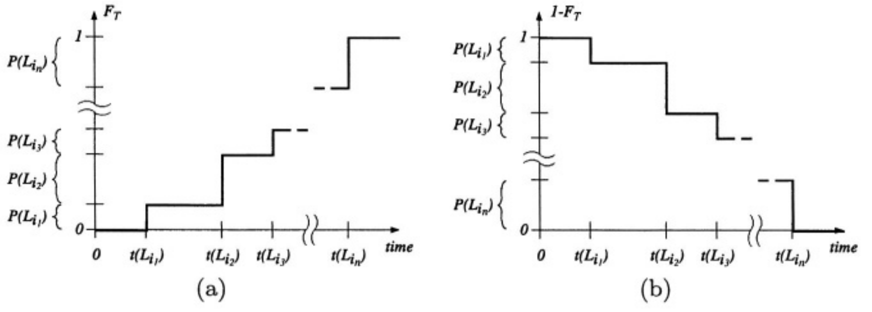


Fig. 2. Defined cumulative distribution functions. (a) F_T (b) $1 - F_T$

In our problem, every valid trajectory R defines a particular cumulative distribution function of finding the object, F_T. Since we are dealing with a discrete problem, the distributions are only piecewise continuous with the discontinuities being the times at which the robot reaches the distinct locations along the route, as shown in Fig. 2a.

By (4), we know that the expected value of a random variable with distribution F_T is the area under the curve $1 - F_T$, shown in Fig. 2b. This area is the value we want to minimize.

One method for making this area small is to have the time intervals as small as possible and the probability changes (down step) as large as possible. This is the notion that our utility function in (3) captures; its value is larger when the probability of seeing the object from a particular location is high (large down step) and/or when the location is near (small time interval).

3.3 Efficient Utility Greedy Algorithm

The utility function in (3) is sufficient to define a 1-step greedy algorithm. At each step, simply evaluate the utility function for all available locations and choose the one with the highest value. This algorithm has a running time of $O\left(n^2\right)$. However, it might be convenient to explore several steps ahead instead of just one to try to "escape local minima" and improve the quality of the solution found. The downside of this idea is that it usually increases the complexity of the algorithm by a factor of $O(n)$ for each look ahead step.

To reduce this effect we propose a second heuristic that reduces the branching factor. The heuristic is that the children of each location can only be those other locations that are not strictly dominated according to the two variables in the utility function $P\left(L_k\right)$ and $Time\left(L_j, L_k\right)$. As seen from the j-th location L_j, a location L_k strictly dominates another L_l if both of the following conditions are true

$$P\left(L_k\right) > P\left(L_l\right),$$
$$Time\left(L_j, L_k\right) < Time\left(L_j, L_l\right).$$

It is straightforward that dominating locations will lie on the convex hull of the remaining set of locations when plotted on the probability vs. distance plane. The endpoints of this partial convex hull are not considered as candidates since they are not defined locations.

The final algorithm for a single robot is as follows:

(1) For the last location along the current solution (initially just the robot starting location) explore the possible routes (create a tree breadth-first) until the number of nodes is of order $O(n)$.
(2) For each node that needs to be expanded, compute the set of locations that are not strictly dominated by others and only choose those as children. This can be done with a convex hull algorithm with complexity $O(n \log n)$.
(3) When the number of nodes in the exploration tree has reached order $O(n)$, choose the best leaf according to the heuristic in (3), discard the current tree and start over with the best node as root.

The complexity of the algorithm is proportional to exploring a tree of order $O(n)$, choosing the best children for each node in the tree with a convex hull algorithm in $O(n \log n)$ and repeating $\frac{n}{\log n}$ times to generate a complete route. This is

$$O\left(n \cdot n \log n \cdot \frac{n}{\log n}\right) = O\left(n^3\right).$$

Of course, this result depends on the number of dominating locations being significantly smaller than n on average, which may be difficult to determine for a specific problem. We know, for example, that the expected number of points on the convex hull of a set sampled uniformly from a convex polygon is of order $O(k \log n)$ for a k-sided polygon [2]. In the worst case, when the branching factor is not reduced at all, our algorithm only explores one step at a time and has a running time of

$$O(n \cdot n \log n \cdot n) = O\left(n^3 \log n\right). \tag{5}$$

3.4 The General Multi-robot Case

For the case of a single robot, each sensing location determines a state for the environment search. Each node in the search graph corresponds exactly to one location. For the case of multiple robots, the state has to be augmented to include the status of every robot in the team. Each robot can be performing one of two possible actions: sensing or traveling. Therefore, a node in the search graph now corresponds to a n-tuple of robot actions. We assume that two or more robots will never arrive to sensing locations exactly at the same time, which is true for sensing locations that are in general position. For example, the n-tuple (T_4, S_{17}, T_8) represents a state in which the first robot (first location in the tuple) is traveling towards location L_4, the second robot is sensing at location L_{17} and the third robot is moving to location L_8.

This approach allows us to discretize time into uneven intervals bounded by critical events. These events correspond to the times robots reach given locations

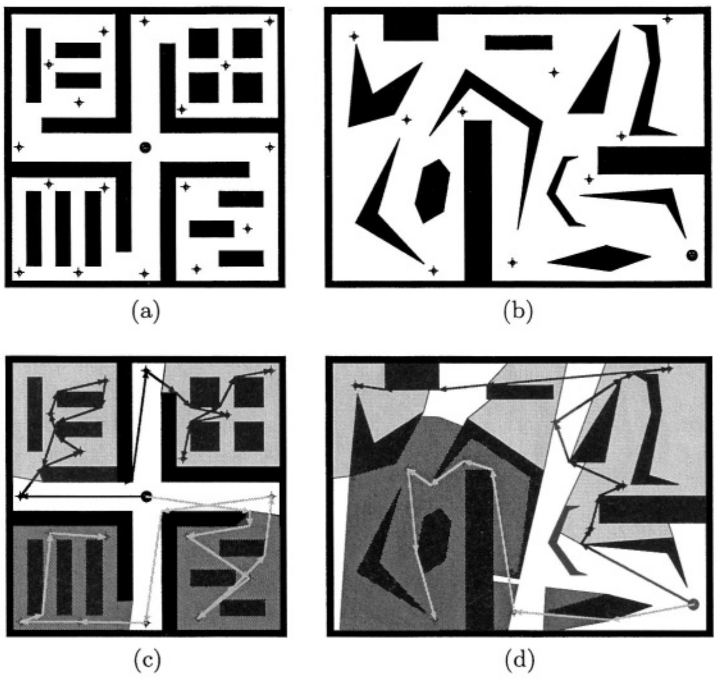


Fig. 3. Simulation results for two environments and a team of two robots

and sense the environment. This formulation defines a new state space which is searched with the same utility function algorithm proposed for a single robot.

Under our scheme, the possible next states only consider the locations already visited by all the robots, and once a robot commits to going to a certain location, it will not change regardless of what the other robots are doing. This yields a scheme which guarantees that one more location will be visited at each exploration step. This means that the computational complexity for a team of robots is exactly the same as for the single robot case established in (5). However, this scheme will not explore all possible path permutations (which would be exponential), only a reduced state space.

3.5 Probability Computation for Polygonal Environments

We assume a uniform probability density function of the object's location over the environment, consequently, the probability of seeing the object from any given location is proportional to the area of the visibility region from that location (point visibility polygon [2]). This visibility polygon can be computed in linear time to the number of vertices in simple polygons and in $O(n \log n)$ time for general polygons [2]. If the results are cached, this has to be done only once for each location.

The probability of seeing the object for the first time at location L_{i_j}, denoted $P(T = t_j)$, is proportional to the area visible from L_{i_j} minus the area already

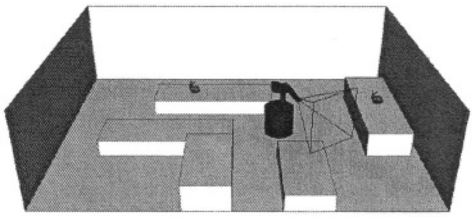


Fig. 4. A 3-D workspace with a mobile manipulator

seen from locations L_{i_k} $\forall k < j$, as stated in (2). This involves polygon clipping operations of union and difference. We know that any clipping algorithm supporting two arbitrary polygons must have a complexity of at least $O(n\ m)$ where n and m are the number of vertices in each polygon [5].

The cost of performing these clipping operations must be added to the complexity in (5) to describe the total complexity of the algorithm when applied to general polygons.

4 Simulation Results

For our simulations, we implemented routines for computing visibility polygons, the reduced visibility graph and shortest paths (Dijkstra's Algorithm). For calculating the union of visibility regions, we used the *gpc* library developed by Alan Murta based on an approach proposed in [16].

Fig. 3 shows the paths generated by our proposed approach for two different environments and a team of two robots. Parts (a) and (b) show the environments (black obstacles), the set of sensing locations (crosses) and the initial position (black circle). Parts (c) and (d) show the final paths and different regions sensed by the robots. The light grey path corresponds to robot 1, and the dark grey to robot 2. The dark grey area was only sensed by robot 1, the light grey only by robot 2 and the white area was seen by both.

As can be seen, the robots split the effort of sensing the environment in approximately equal amounts, even when an intuitive partition is not evident, as in the case of the environment shown in the right column of Fig. 3.

For the case of a single robot, the results obtained by our algorithm are close to the optimal case but with a major improve in computation time [12]. In some instances, we have observed over a thousand-fold improvement. For a team of multiple robots, the expected value of the time is further reduced, as expected, *but with the exact same order of computational complexity.*

5 Conclusions and Future Work

In this paper we proposed an efficient approach to solve the problem of finding an object in a polygonal environment as quickly as possible on average. We proposed the heuristic of a utility function to drive a greedy algorithm in a reduced search space that is able to explore several steps ahead without incurring too high a computational cost. We implemented this algorithm and presented simulation results for a multi-robot scheme.

In [13] we address the problem of continuous sensing for expected value search with a single robot. For this, we use the calculus of variations to compute locally optimal sub-paths and concatenate them to construct complete trajectories. As future work, we plan on extending the continuous sensing case to multiple robots.

Currently, we are also working on the case where the environment is three dimensional and the robot is a seven degree of freedom mobile manipulator with an "eye-in-hand" sensor, like the one shown in Fig. 4.

References

1. Chin, W.P. and S. Ntafos, "Optimum Watchman Routes," *Information Processing Letters,* Vol. 28, pp. 39–44, 1988.
2. Goodman, J.E. and J. O'Rourke, *Handbook of Discrete and Computational Geometry,* CRC Press, 1997.
3. Garey, M.R. and D.S. Johnson, *Computers and Intractability,* W. H. Freeman and Company, 1979.
4. González-Baños, H.H. and J.C. Latombe, "Navigation Strategies for Exploring Indoor Environments," *Int. Journal of Robotics Research,* 21(10/11), pp. 829–848, Oct–Nov 2002.
5. Greiner, G. and K. Hormann, "Efficient Clipping of Arbitrary Polygons," *ACM Transactions on Graphics,* Vol. 17, number 2, pp. 71–83, 1998.
6. LaValle, S.M. *et al,* "Finding an Unpredictable Target in a Workspace with Obstacles," *in Proc. IEEE Int. Conf. on Robotics and Automation 1997.*
7. Murrieta-Cid, R., A. Sarmiento and S. A. Hutchinson, "On the Existence of a Strategy to Maintain a Moving Target within the Sensing Range of an Observer Reacting with Delay," *in Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems 2003.*
8. Murrieta-Cid, R., A. Sarmiento, S. Bhattacharya and S.A. Hutchinson, "Maintaining Visibility of a Moving Target at a Fixed Distance: The Case of Observer Bounded Speed," *in Proc. IEEE Int. Conf. on Robotics and Automation 2004.*
9. O'Rourke, J., *Art Gallery Theorems and Algorithms,* Oxford University Press,1987.
10. Rohnert, H., "Shortest Paths in the Plane with Convex Polygonal Obstacles," *Information Processing Letters,* 23:71–76, 1986.
11. Ross, S.M., *Introduction to Probability and Statistics for Engineers and Scientists,* Wiley, 1987.
12. Sarmiento A., R. Murrieta and S.A. Hutchinson, "A Strategy for Searching an Object with a Mobile Robot," *in Proc. Int. Conf. on Advanced Robotics 2003.*
13. Sarmiento A., R. Murrieta and S.A. Hutchinson, "Planning Expected-time Optimal Paths for Searching Known Environments," *accepted to IEEE/RSJ Int. Conf. on Intelligent Robots and Systems 2004.*
14. Shermer, T.C., "Recent Results in Art Galleries," *in Proc. of the IEEE,* Vol. 80, issue 9, September 1992.
15. Tovar, B., S.M. LaValle and R. Murrieta-Cid, "Optimal Navigation and Object Finding without Geometric Maps or Localization," *in Proc. IEEE Int. Conf. on Robotics and Automation 2003.*
16. Vatti, B.R., "A Generic Solution to Polygon Clipping," *Communications of the ACM,* 35(7), pp. 56–63, July 1992.

Internet-Based Teleoperation Control with Real-Time Haptic and Visual Feedback

Fernando D. Von Borstel and José L. Gordillo

Tecnológico de Monterrey, Centro de Sistemas Inteligentes,
E. Garza Sada 2501, 64849 Monterrey N.L. México
{A00777172, JLGordillo}@itesm.mx
http://www-csi.mty.itesm.mx

Abstract. The real-time control proposed in this paper combines the event-based control theory with numeric potential field and computer vision techniques to teleoperate, via the Internet, a nonholonomic robot. The user receives visual and haptic sensory information in real time, allowing obstacle avoidance while navigating a remote environment. A numeric grid generated from a top view of the workspace is used to produce virtual forces around the obstacles. A computer-based vision system recognizes landmarks at the top of the robot to obtain its current position. The robot position is combined with the grid to generate real-time haptic interaction. A graphic landmark on the workspace image represents the predicted and current robot position. The system was tested experimentally to validate the approach using an Internet2 connection.

1 Introduction

Providing the user with sensory information in real time is always desirable to increase certainty when performing Internet-based teleoperation. Visual feedback is common in most of the Internet-based telerobotics systems reported in the literature [1–3]. Recent Internet-based teleoperation applications, using an event-based controller design introduced in [4], successfully sent sensory information in real time dealing with unpredictable packet delays, lost packets, and disconnection [5,6]. A real-time control for Internet-based teleoperation with force reflection using the event-based controller was presented in [7]. This teleoperation system controls a holonomic mobile robot using information from sonar sensors to generate *virtual forces* that correspond to a robot status in the remote environment.

This paper proposes a real-time control for Internet-based teleoperation of a nonholonomic differentially-driven mobile robot using a visual sensor to provide sensory information in real time. The teleoperation system makes use of the visual sensory information to generate haptic and visual feedback that corresponds to a robot status in the workspace. The implemented teleoperation system was integrated into a prior mobile robotics Virtual Laboratory (VL) framework [8] as another operation mode. The VL teleoperation mode allows the user to control the robot to navigate a small workspace avoiding obstacles. The system combines a modified real-time event-based controller with computer vision and potential field techniques to control the robot. The user indicates the obstacles' vertices on a workspace image and the system computes the robot's *C-Space*. This *C-Space* is converted into a numeric potential field grid where the cell values represent spatial information in a similar manner as the occupancy grids [9]. This numeric grid defines three navigation spaces, as shown in Fig. 1. The spaces have cells with different probability values of being occupied by an obstacle *P(obs),*

$$P(obs) = \begin{cases} 1 & \text{forbidden space} \\ \frac{1}{f(d)} & \text{constrained space} \\ 0 & \text{free space} \end{cases} \tag{1}$$

where $f(d)$ is a linear function of the distance d between the robot and the forbidden space (the obstacle). A computer vision system calculates the robot position in real time to be used as reference to perform several potential field measures on the numeric grid. These measures are combined to generate a repulsive *virtual force* around the obstacles. The system provides real-time visual information regarding the current robot's position in the remote environment by drawing a graphic mark on the workspace images shown by the user interface.

This document describes the proposed real-time control to teleoperate the nonholonomic robot, as well as the system implementation and the experimental results. The article is organized as follows: Section 2 gives a detailed description of the real-time control, Section 3 describes the system implementation and the experimental results using an Internet2 connection, and conclusions are presented in Section 4.

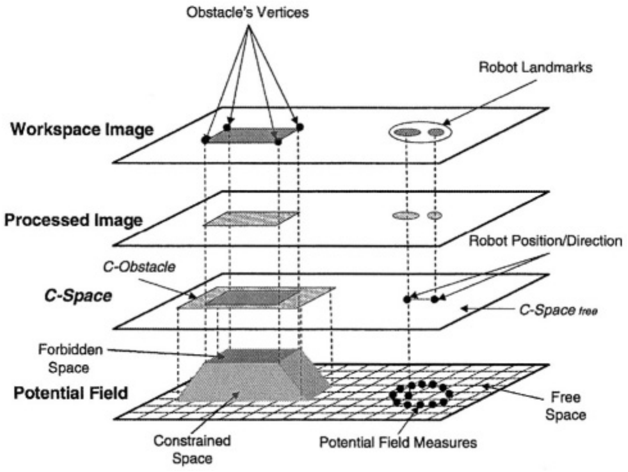


Fig. 1. The numeric potential field takes as reference the *C-Space*

2 The Real-Time Control

The numeric potential field is set when the user indicates the physical obstacle vertices on a top view image of the workspace. These vertices are used to compute a *C-Space.* Then the *C-Space* is converted into a numeric potential field grid, which is used to generate a repulsive *virtual force* when the robot approaches to an obstacle. The *virtual force vector* is obtained based on the proximity and direction to the obstacles, which are calculated from several potential field measures taken from the numeric grid using as reference the current robot's position obtained from the computer vision system. This *virtual force* is converted to a velocity value and introduced into the real-time control that generates a velocity tracking error. This tracking error is fed to the user as a force feedback to indicate the proximity and direction to the obstacles. Furthermore, a graphic mark is also drawn in real time on the image to indicate the robot's position.

2.1 System Model

The system architecture on the Internet was based on two elements: the first element, called *Guest,* is a computer connected to the Internet containing one or more user interfaces that allow the user teleoperate the slave device; a Khepera minirobot [10]. In contrast, the other element called *Host,* is a computer connected to the Internet containing the necessary applications to remotely operate the slave device. The *Guest* presents a user interface that shows a workspace image and is connected to a joystick. The robot and a camera are connected to the *Host,* as shown in Fig. 2. Each block will be described in detail and all variables are defined in Table 1 and referenced to s, where s is the event, $s \in \{1,2,..n\}$, and represents the number of commands issued by the user. The stability and synchronization of the system result from the use of a non-time based variable s as reference [11].

Image Acquisition. A top view of the workspace is acquired by the camera and converted into a discrete image using a video frame grabber. This process generates a matrix $I(s)$ of integer elements using a 256 level grayscale. The matrix $I(s+n)$ is compressed using the Run Length Encoding (RLE) method before be sent to the potential field creation procedure.

Potential Field Creation. The image matrix $I(s+n)$ is decompressed and displayed on the user interface. Next, the user interacts with the user interface to define a *C-Space* based on the workspace image. Using the computer mouse, the user indicates the obstacles' vertices on the image and the system calculates and draws a convex polygon that is expanded in relation with the robot radius, to build a *C-Obstacle.* This procedure is carried out m times, once per each physical obstacle. The system then converts the displayed *C-Space* bitmap into a numeric matrix M with the same size than $I(s)$, where the obstacles are expanded in a decremented way to create a numeric potential field. The numeric matrix M is calculated once.

Image Processing. The matrix $I(s)$ is binarized and segmented to find every object in the image. Then the objects are characterized using Hu invariant

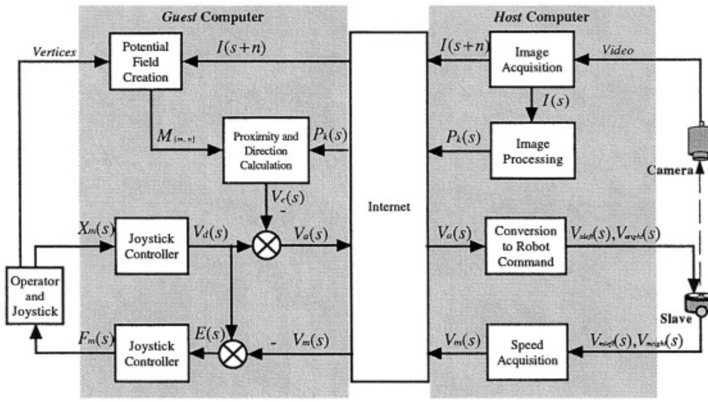


Fig. 2. Block diagram of the real-time control for Internet-based teleoperation

Table 1. Definition of the Variables used in the Block Diagram

Variable Definition	Concept
$F_m(s) = [F_{mx}(s)F_{my}(s)]^T$	Applied force by Joystick
$X_m(s) = [X_{mx}(s)X_{my}(s)]^T$	Joystick position
$V_d(s) = [V_{dleft}(s)V_{dright}(s)]^T$	Desired velocity
$V_a(s) = [V_{aleft}(s)X_{aright}(s)]^T$	Applied velocity
$V_m(s) = [V_{mleft}(s)V_{mright}(s)]^T$	Measured velocity
$E(s) = [E_x(s)E_y(s)]^T$	Tracking error
$V_e(s) = [V_{eleft}(s)V_{eright}(s)]^T$	Potential field velocity
$P_k(s) = [X(s)Y(s)\theta(s)]^T$	Robot position

moments to identify a robot's artificial landmark. Two circles compose the robot's landmark: the bigger circle represents the robot's front, while the small one indicates the rear. The robot's position $P_k(s)$ is the centroid of the big circle. The robot orientation is the angle defined by a line passing through the centroids of both circles and the x-axis of the reference frame.

Operator and Joystick. The joystick and the operator have a spring-like behavior [12]; the operator generates new joystick positions to compensate the displacements generated by the force feedbacks, according to the following function:

$$X_m(s) = \frac{F_m(s-1)}{K_m}, \tag{2}$$

where K_m is a scaling constant, the new joystick position is $X_m(s)$ and the previously played force is $F_m(s-1)$. Take notice that the rotational component $X_{m\theta}(s)$ is not used, since the joystick is not able to feedback force in that direction.

Joystick Controller. The position $X_m(s)$ is proportionally converted to the desired velocity $V_d(s)$ for both left and right robot motors allow for differential

Table 2. Equations for Joystick Position Conversion to Desired Velocity

$X_{mx}(s), X_{my}(s)$	V_{dleft}	V_{dright}
$+X_{mx}(s), -X_{my}(s)$	$-\frac{X_{my}(s)}{K_p}$	$\frac{-X_{my}(s)-\frac{X_{mx}(s)}{K_x}}{K_p}$
$+X_{mx}(s), +X_{my}(s)$	$-\frac{X_{my}(s)}{K_p}$	$\frac{-X_{my}(s)+\frac{X_{mx}(s)}{K_x}}{K_p}$
$-X_{mx}(s), -X_{my}(s)$	$\frac{-X_{my}(s)+\frac{X_{mx}(s)}{K_x}}{K_p}$	$-\frac{X_{my}(s)}{K_p}$
$-X_{mx}(s), +X_{my}(s)$	$\frac{-X_{my}(s)-\frac{X_{mx}(s)}{K_x}}{K_p}$	$-\frac{X_{my}(s)}{K_p}$
$+X_{mx}(s), 0$	0	$-\frac{X_{mx}(s)}{K_p \times K_x}$
$X_{mx}(s), 0$	$-\frac{X_{mx}(s)}{K_p \times K_x}$	0

drive. The $X_m(s)$ and equations in Table 2 are used to obtain the velocity $V_d(s)$. The equations in the four first rows use K_x to decrease the component $X_{mx}(s)$ to generate a hyperbolic speed function on the x-axis of the joystick reference frame. K_p is a scaling constant, e.g. K_x =3, K_p=100.

Proximity and Direction Calculation. Potential field measurements are performed taking as reference the robot position $P_k(s)$. These measures are treated as proximity sensor readings to generate *virtualforces* which are converted to velocity values to be fed into the real-time control. A punctual measure over the numeric potential field is called *virtual sensor* $M_n(s)$. Six *virtual sensors* are distributed around the robot's front (as the real ones) and another six are placed around the robot's rear. The *virtual sensors* make proximity measures on the numeric potential field grid around the obstacles. These measures are condensed by equations 3 and 4 in two variables: proximity $P_r(s)$ and direction $\phi(s)$ [13]:

$$P_r(s) = \lfloor \frac{Max(M_0(s), M_1(s), M_2(s), M_3(s), M_4(s), M_5(s))}{K_s} \rfloor, \tag{3}$$

$$\phi(s) = \lfloor \frac{90(M_5(s) - M_0(s)) + 45(M_4(s) - M_1(s)) + 5(M_3(s) - M_2(s))}{9(1 + \sum_{i=0}^{5} M_i(S))} \rfloor. \tag{4}$$

The variable $P_r(s)$ is calculated based on the maximum proximity measure which is divided by a scaling constant K_s. The variable $\phi(s)$ is computed subtracting the opposite measures of the *virtual sensors* from the corresponding left or right sides, therefore a positive or negative sign is obtained indicating the left or right side of the robot's front/rear. The results from the subtraction are multiplied by weights corresponding to the location of the *virtual sensors.* The $\phi(s)$ value is obtained summing the weighted subtraction results. Then $\phi(s)$ is divided by a summation to normalize the results into a –10 < 0 < +10 range. The $P_r(s)$ and $\phi(s)$ behavior is described in detail by [13].

When the robot performs a forward displacement, the six frontal *virtual sensors* are taken into account to calculate $P_r(s)$ and $\phi(s)$ and generate velocity $V_e(s)$ according to the potential field. When a backward displacement is performed the six rear *virtual sensors* are taken into account to calculate $P_r(s)$ and

Table 3. Equations for Proximity Conversion to Potential Field Velocity

$\phi(s)$	V_{eleft}	V_{eright}
$\phi(s) = 0$	$\frac{P_r(s) \times V_{dleft}(s)}{K_e}$	$\frac{P_r(s) \times V_{dright}(s)}{K_e}$
$\phi(s) > 0$	0	$\frac{P_r(s) \times V_{dright}(s) \times cos(10\phi(s))}{K_e}$
$\phi(s) < 0$	$\frac{P_r \times V_{dleft}(s) \times cos(-10\phi(s))}{K_e}$	0

$\phi(s)$ and generate $V_e(s)$. The velocity $V_e(s)$ is calculated based on the equations on Table 3. In these equations, $P_r(s)$ represents the *virtual force vector* magnitude and $\phi(s)$ is used to obtain the $P_r(s)$ vector projection, on the corresponding axis, to generate the velocity $V_e(s)$ on both left and right motors, using K_e as a scaling constant. After that, the calculated $V_e(s)$ is subtracted to $V_d(s)$ to produce an applied velocity $V_a(s)$ to the robot:

$$V_a(s) = V_d(s) - V_e(s). \tag{5}$$

Conversion to Robot Command. The velocity $V_a(s)$ is converted into a command string recognized by the robot speed mode [10], with the corresponding parameters to set both left and right wheel motor speeds $(V_{aleft}(s), V_{aright}(s))$.

Slave (Mobile Robot). The robot receives the speed command string and executes and holds the command for a specific time frame, then stops and waits for the next command. The speed commands set each wheel motor speed through the robot PID controller and a PWM (Pulse Width Modulation) generator [10].

Speed Acquisition. Once applied the velocity $V_a(s)$ to the robot, the speed acquisition procedure requests the instantaneous speed [10] of both wheel motors $(V_{mleft}(s), V_{mright}(s))$, to be sent as the measured velocity $V_m(s)$ to the *Host* system. $V_d(s)$ is taken as a reference, the difference between $V_d(s)$ and $V_m(s)$ generates a tracking error $E(s)$ that is converted to a force vector $F_m(s)$ by the joystick controller.

$$E(s) = V_d(s) - V_m(s). \tag{6}$$

The joystick controller converts the error $E(s)$ into the $F_m(s)$ applying the inverse procedure of Table 2 equations to obtain the error projection $E(s) = [E_x(s)E_y(s)]^T$ on the x and y axes. This vector $E(s)$ is added to the last applied force $F_m(s-1)$. The resulting vector $F'_m(s)$ is multiplied by a scaling constant K_f to obtain the current force vector $F_m(s)$. Then, the $F_m(s)$ vector is played by the joystick during a specific time frame:

$$F'_m(s) = F_m(s-1) + E(s), \tag{7}$$

$$F_m(s) = K_f F'_m(s). \tag{8}$$

Internet. The real-time control takes the communication link as a delay element that plays no role in the control model [7].

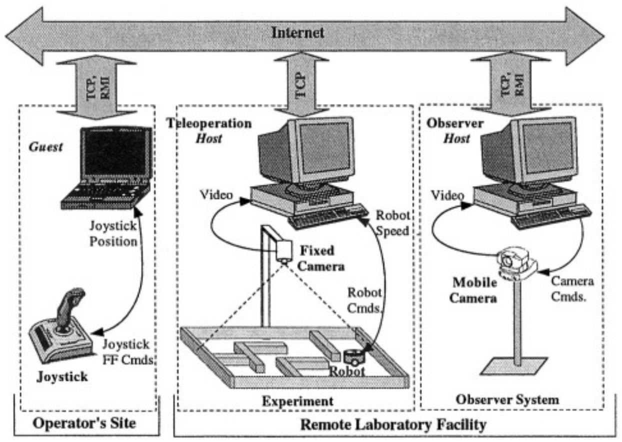


Fig. 3. Teleoperation system configuration to navigate a small workspace

3 System Implementation

This section describes the experimental setup of the teleoperation system, which is complemented with an independent observer system described in [8]. All user interfaces were run in a *Guest* computer and the server applications were executed in two *Host* computers: one computer controlled the robot, and the other served as the observer. Fig. 3 shows the experimental configuration. The workspace, the *Guest* and the *Host* computers have the same setup described in [8]. The robot motor speeds ranged from ±10 pulses/10 ms and are held for 90 ms. On the other side, the force feedback is provided by the joystick for 250 ms. The joystick is a MS Sidewinder FF2 connected to a USB (Universal Serial Bus) port of the *Guest* computer. The joystick controller was programmed in C++ including MS DirectX libraries, and it communicates via TCP (Transport Control Protocol) with the *Guest* application written in Java. Left Fig. 4 shows the teleoperation interfaces, which have drawn the *C-Space* used in the experimentation, and the graphic robot mark that represents the current robot position. The graphic mark is a triangle that shows the robot's direction and position. The triangle is drawn inside a dotted circle. These dots represent the virtual sensors around the robot and indicate its position. A dotted line is drawn indicating the predicted position of the robot based on the last applied velocity. The communication link is performed using TCP for the teleoperation system. The number of compressed images sent by the teleoperation *Host* computer is reduced to avoid unnecessary delays introduced by the image transmission through the Internet. An image is sent to the *Guest* every n events (operator commands). Consequently, every $s + n$ event an image is transmitted and displayed. The compressed images range from 16 to 19 Kbytes.

4 Experimental Results

This section describes one of several performed experiments. The results were acquired using an Internet2 connection with 10 jumps in the route, according traceroute's output. The *Guest* was located at CIBNOR (Centro de Investigaciones Biológicas del Noroeste La Paz, Mexico) 1,050 km away from the *Host* system, which was situated in the Robotics and Vision Laboratory of the CSI (Centro de Sistemas Inteligentes, ITESM campus Monterrey, Mexico). Right Fig. 4 shows the robot's trajectory on the (x, y) plane; it is 94.710 cm long. The potential field around the obstacles is represented by boundary lines and the *virtual sensor* positions during the trajectory are indicated by the scattered points. The trajectory plot has a resolution of 0.30625 cm per point. The trajectory was performed in 163.445 sec. The average time for a complete system cycle was 0.9080 sec. Fig. 5 presents the experiment behavior; the plots are referenced to the event s; 180 events (commands) were performed. The first row presents the desired velocity $V_d(s)$ in pulses/10 ms versus s, for the left and right motors of the robot. The second row shows the sum of the measured velocities $V_m(s)$ and the corresponding y-axis force $F_{my}(s)$ versus s. It is clear that the plots are similar, considering that the sum of the measured velocities and the force felt in the y-axis should behave as the desired velocities. The third row illustrates the force applied by the joystick in both x and y axes. The desired velocity $V_d(s)$ is decremented according to $P_r(s)$ and $\phi(s)$ which are computed using the measures of the *virtual sensors* on matrix *M*. If an obstacle is near the robot's front, then both left and right motor velocities decrease, an inverse force vector increases in the y-axis of the joystick, the robot's rear has the same behavior. If an obstacle is near the right side of the robot's front, then the right motor velocity decreases, but a negative force in the x and y axes of the joystick increase; indicating that an obstacle is near in that direction, and for the left side, a negative force in the y-axis and a positive force in the x-axis is generated, and vice versa for the robot's rear. Thus a closer

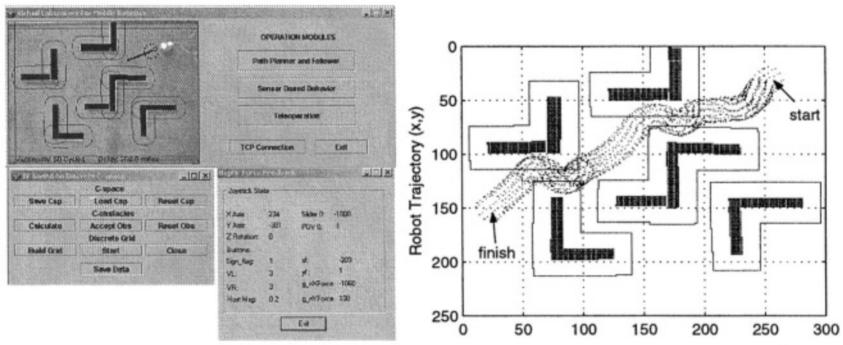


Fig. 4. Left figure presents the user Interfaces for teleoperation showed in the *Guest* computer. Right figure shows the robot's trajectory on the (x, y) plane

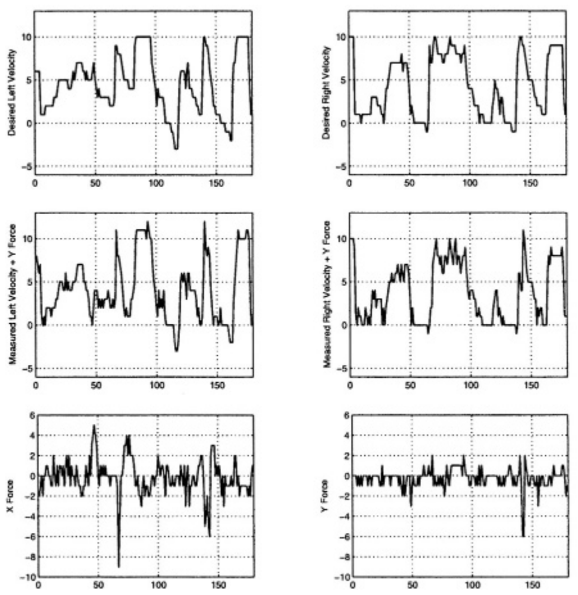


Fig. 5. The system behavior during a remote experiment

obstacle causes the tracking error to increase, and this error is proportionally converted to a force vector by the joystick, to tell the user that an obstacle is present in the robot trajectory. If the robot is located at 3.675 cm from the obstacle, then the left and right motors stop to avoid collision. When the *virtual sensors* have not detected any obstacle around the robot, the measured velocity starts tracking the desired one, and both the tracking error and force vector decrease. The tracking error variation is influenced by the tracking error of the robot's PID controller, by the friction of the surface, and by the weight of the serial cable attached to the robot. The most significance peaks shown on the y-axis force plot are due to the presence of obstacles in the robot trajectory.

5 Conclusions

This paper presented a real-time control system for Internet-based teleoperation of nonholonomic differentially-driven robots. The described control model generates real-time sensory information using potential field and computer vision techniques, in conjunction with an event-based controller. The implemented system allows to teleoperate a mobile minirobot and provides the user with haptic and visual feedback in real time that enhances the control experience by making it more natural and intuitive. Results obtained using an Internet2 connection validate the design of this control model.

References

1. Michel, O., Saucy, P., Mondada, F.: KhepOnTheWeb: an experimental demonstrator in telerobotics and virtual reality. IEEE Robot. & Autom. Magazine **7** (March 2000) 41–47
2. Hirukawa, H., Hara, I.: Web-top robotics using the world wide web as a platform for building robotic systems. IEEE Robot. & Autom. Magazine (June 2000) 40–45
3. Taylor, K., Dalton, B.: Internet robots: a new robotics niche. IEEE Robot. &; Autom. Magazine (March 2000) 27–34
4. Xi, N.: Event-based planning and control for robotics systems. Doctoral dissertation,Washington University (december 1993)
5. Lo, W. T., Liu, Y. H., Xi, N., Shi, Y., Wang, Y.: Co-operative control of internet based multi-robot systems with force reflection. Proc. Intl. Conf. on Robot. & Autom. (September 2003)
6. Elhajj, I., Xi, N., Fung, W. K.,Liu, Y. H., Hasegawa, Y., Fukuda, T.: Supermedia-enhanced internet-based telerobotics. Proc. of the IEEE **91** (March 2003)
7. Elhajj, I., Xi, N.,Liu, Y. H.: Real-time control of internet based teleoperation with force reflection. Proc. IEEE Intl. Conf. on Robot. & Autom. (April 2000) 3284–3289
8. Von Borstel, F. D., Ponce, B. A., Gordillo, J. L.: Mobile robotics virtual laboratory over the internet. Proc. Fourth Mexican Intl. Conf. on Computer Science, ENC 2003, IEEE Computer Society Press (September 2003) 308–314
9. Elfes, A. : Using occupancy grids for mobile robot perception and navigation. IEEE Computer **22** Issue 6 (june 1989).
10. K-Team: Khepera user manual. Ver. 5.02 (1999), http://www.k-team.com
11. Xi, N., Tarn, J.: Action synchronization and control on internet based telerobotics systems Proc. IEEE Intl. Conf. on Robot. & Autom. **1** (1999) 219–224
12. Nogan, N.: Multivariable mechanics of the neuromuscular system. Proc. IEEE Eight Annual Conf. of Engineering in Medicine and Biology Society (1986)
13. Diard, J., Lebeltel, O.: Bayesian learning experiments with a khepera robot. Proc. First Intl. Khepera Workshop HNI-Verlagsschriftenreihe (1999) 129-138

Representation Development and Behavior Modifiers

Carlos R. de la Mora B.[1], Carlos Gershenson[2], and V. Angélica García-Vega[3]

[1] AI-Lab, Vrije Universiteit Brussel, Building G 10 room 725. Pleinlaan 2. 1050 Brussels - Belgium
carlos@arti.vub.ac.be

[2] CLEA, Vrije Universiteit Brussel, Krijgskundestraat 33, B-1160 Brussels - Belgium
cgershen@vub.ac.be

[3] Facultad de Física e Inteligencia Artificial, Universidad Veracruzana, Sebastián Camacho No. 5, Xalapa, Ver., México 91000
angegarcia@uv.mx

Abstract. We address the problem of the development of representations by an agent and its relationship to the environment. A software agent develops a representation of its environment through a network, which captures and integrates the relationships between agent and environment through a *closure mechanism.* A variable behavior modifier improves the representation development. We report the preliminary results where we analyze two aspects: 1) The *structural properties* of the resulting representation can be used as indicators of the knowledge assimilated by the agent from the interaction with the environment. These properties can be taken as useful macrovariables from an objective point of view; and 2) The dynamics of the closure mechanism, can be seen as the internal, and therefore subjective, way used by the system to develop its representation. We are not interested only on how the mechanism functions, but also on how the representation evolves.

Keywords: Closure, representation development, behavior modifiers, affective states, biological motivations.

1 Introduction

Any cognitive agent must be able to start, distinguish, improve, and represent actions in an autonomous manner. Language can be seen as intersubjective actions and not just a code-decode system, therefore language can be seen, also, as an embedded use-action activity with an implicit meaning. How can actions be represented internally, how can they be evolved, discovered and improved? Answers to these questions can be seen as part of a process under a constructivist or epigenetic approach. Under this approach, embodiment, social interaction, development and integration are important specific topics, as Brooks has pointed [1]. Embodiment is a property any cognitive agent must have. It is associated with the representation of the world, the body of the agent interacting with the environment outlines, influences, limits those representations. Some Artificial Intelligence researchers have recognized the need of representations to have a truly situated and cognitive agent [14]. Mitchell has pointed the need to integrate the dynamical and state approaches to representation also has to be related with the dynamics of the system agent-environment[8]. Thus, we are interested in the dynamics of the interactions generating representations, and therefore creating structure. Ziemke [19] has proposed five notions of embodiment. Each of them has distinctive significance and relevance for the epigenetic phenomena. We are interested in the *organismic embodiment* of autopoietic living systems, which is based on the idea that cognition is what living systems do when they interact with their environment (and other organisms) [7], [16]. This kind of embodiment includes development in all biological ways of organization. Obviously, there is a clear difference between living organisms, which are autonomous and autopoietic, and man-made machines, which are heteronomous and allopoietic. So, how can an artificial autopoietic system be built?

Steels has proposed that representations can be explicit (or symbolic) but also implicit (or emergent) [14]. Zlatev and Balkenius [18] have proposed that any epigenetic robot must have the ability to generate representations, but it is not clear what internal representations are and how are they developed in living systems. We assume that any external representation requires an internal representation to acquire meaning. We think that internal representations are the intertwined relationships between significant perceptions and significant actions of an embodied agent interacting within an environment.

With this background, in this work we try to relate the dynamic and structural descriptions in representation development. The paper is organized as follows: Section 2 defines the pragmatic games used to explore representation emergence, describes how the pragmatic games were implemented and which are the components of the software agent. Section 3 defines the concepts of closure, closure states and describes how they are generated. It also defines behavior modulators and presents the type of behavior modulator: focus, used to improve representation development. In section 4 we present two perspectives to analize the experiments carried on: internal description and the external description. At last, section 5 closes the paper with discussion and future work.

2 Pragmatic Games and Experimental Setup

To develop and analyze representations, we need a setup in which the agent copes with enough similar situations to construct knowledge, as Piaget thought [9]. The easiest way to do this is by interacting within an environment, one simple enough to provide similar (but not identical) conditions. "Pragmatic games" can be played to achieve this scenario: every time an event occurs, the scenario restarts with similar conditions. The term "pragmatic games" is inspired on "language games" [13], but we used them, also, as a methodology to study epigenetic development. The agent can be carried out pragmatic games as a result of its capabilities and the environment's characteristics. It means that the agent has the possibility to play the game and complete it with no more than the inborn capabilities. The agent can move allowing errors.

The pragmatic game used to contrast our ideas is the "feeding game", a subset of the micro-world used by Drescher [5] to study Piaget's Schemes. This involves a 2D 7x7-grid world in which there is an agent consisting of one 5 × 5-grid "eye" with a central 1×1 square "fovea", a 1 × 1 "hand", and a 1 × 1 "mouth". Within the world "objects" of size 1×1 can exist. The agent has four independent *actuators,* to move its hand and eye in the two spatial dimensions. The eye's movements are restricted to focusing of the fovea within the world. The hand has the same constraint. The displacements are discrete and equal in length side of one cell, in such a way that the eye and the hand always occupy complete world cells. Each of the four actuators chooses randomly among three possible options: decrease, maintain, or increase (-1, 0, or 1) the actual positions of the eye and hand in both dimensions (e_x, e_y, h_x, h_y), constrained by the environment. Each value has equally probability to occur. An *actuation* would be a set of four values of the actuators. In the environment, an object can be placed randomly anywhere. If the hand passes over the object, it will be attached to the hand. If the hand holding the object passes over the mouth, the object is "fed", and a new object appears at a random location, and the game starts again. Each one of the 25 cells of the eye senses the colors R, G, or B of the objects in the visual field, sending 3 bits to the agent, one for each color. Therefore, the eye's sensing signal consists of 75 bits. The "hand" and the "mouth" contribute to the total sensing vector with one bit each if their position coincides with an object. The agent has no proprioception, in the sense that it has no register of the relative position of its hand, eye, nor mouth. In addition to the sensing states, we define a set of distinguishable *innate biological motivations,* the incoming information consisting of a 5-bit vector: Three bits for the fovea, each one for detecting R, G or B, one for the "hand" and other for the "mouth". Therefore, there are potentially 32 different biological motivations, although in our simple simulations less than ten are bootstrapped. They only are distinguishable and at the beginning they are not related to any sensorial state.

3 Closure Mechanism

We used the small-world networks theory to study the dynamics and structural properties of the resulting network. The network will be strongly related to the particular choice of the *closure mechanism* because this affects how nodes and arcs are introduced to the network [15]. We consider a *closure mechanism* as a mechanism that favors probabilistically category formation]] [6].

In our experiments, the initial representation network is empty. The agent has as inputs the *sensing states* and *biological motivations.* Every time the agent experiences a particular biological motivation, a record is created saving the *sensing states* associated with it.

Any distinguishable situation is considered as a *signal* and the agent has to incorporate it in the representation. In a simplified way, a process is *closed* if an *actuation* is related with the *signals* and the *signals* are related with the *actuation.* In this sense, the *closure mechanism* must be a process *trying to introduce relevant signals and actuations in the representation and trying to identify if they are or not related.* In our directed graph, the nodes will have the *signal* information and the arcs the *actuation* information.

The *closure mechanism* will create and incorporate relevant nodes and arcs, modifying their status. When it reaches a class of "well formed" links between nodes, they will be called *facts,* having some relation with the schemas of Drescher [5] but constructed with different criteria and motivations[1].

Nodes which are *affective states* [11] are added by the agent to develop the representation. *Affective states* represent any state of the agent that could affect it, for better or worse. For our aim, this distinguishable character is indispensable to filter information from the world and to establish organization, measurements does not have this characteristic. *Biological motivations* are considered as affective states, since they are distinguishable. Under this point of view, *sensing states* are not *affective states,* because sensing has no relevance to the agent. The importance of a *sensing state* requires to be captured into a representation in order to acquire some *value* (relative to the agent).

A *sensing state* can become into an *affective state* if it becomes to associate with some value. An *affective state* can be seen as a signal. These can become related through *actuations,* enabling the agent to develop a structured representation in an autonomous way. The relationship between *signals* and *actuations* is structured even when the actuations are random. This structure is reflected in two ways: internally, during the *closure mechanism,* and externally, analyzing the network properties.

After a certain number of iterations, the system "falls" in a process trying to determine if the *biological motivations* have some specific associated *sensing state.* Therefore the *sensing* bits always present when the *biological motivation* has been experienced are represented in the *affective state,* and; if the *sensing state* at time t corresponds to an *affective state,* then a node corresponding to the *sensing state* in the time $t-1$ is incorporated in the representation, as well the directed arc between the nodes (representing the *actuation*). The new node is called *potential affective state.* A *potential affective state* becomes *affective state* if its frequency exceeds some value[2]. The relationships between nodes (arcs) can be incorporated in the representation in two ways: When a *potential affective state* is created, and; When the agent experiences two *sensing states* have been associated with existing nodes *(affective states* or *potential affective states)* in the representation.

[1] For Drescher a functional relationship is searched, being reliability the last test. For us, only a significant departing from randomness in actuations establishes a deep relationship between affective states, stressing a non functional relationship between them.

[2] If the values are too small, noise can be learned. If the values are too big, then it takes more time to learn. This also happens with other parameters of the model.

Every time an arc is crossed, the frequency and the performed *actuation* are recorded in the arc. Once the arc exists in the network, its status can be modified with the recorded information of the actuations during the process in the following way: 1) If the frequency of occurrence in experiencing a specific arc is larger than a given value, it becomes a *frequent arc* and the probability distributions of the actuators are computed from the history and saved in the arc in the form: $\{\{p(e_x = -1), p(e_x = 0), p(e_x = 1)\}, \{p(e_y = -1), p(e_y = 0), p(e_y = 1)\}, \{p(h_x = -1), p(h_x = 0), p(h_x = 1)\}, \{p(h_y = -1), p(h_y = 0), p(hy = 1)\}\}$; 2) An arc is considered *codifiable* if one of the 12 probabilities of a *frequent arc* is greater than a threshold, since the nodes joining the arc have more than a random link, as the movement could be codified for at least one *actuator.* If a *codifiable arc* has *affective states* as source and target nodes, it will be called a *fact.* This is the most refined state for the closure mechanism in the development of the arcs. This contrasts with the Drescher's perspective, which considers reliability as the way to verify the arc's functionality.

Our method is imperfect associating nodes in a strict causal way, but has an advantage: it has no intention into reach specific nodes or to proof specific arcs, avoiding any "cognitive" consideration. The representation is developed through the imperfect agent's possibilities and not under our preconceived "true or false" considerations. This is because we are interested in the way the closure process occurs and not in its success as being "the best" fact constructor. We stress that the important aspect is the network formed by *facts,* and that the closure mechanism does not stop when an arc has achieved the fact character, but continuously incorporates new arcs and nodes (which can become facts).

3.1 Closure States

To identify the agent's closure state associated to every pair of consecutive *sensing states,* we use a set of three values (according to Table 3.1), one for the sensing in the time $t-1$, other for the sensing in the time t, and the third for the state of the link between them.

Table 1. Codes of closure states

value	node	arc
0	not in representation	not in representation
1	potential affective state	not frequent
2	affective state	frequent
3	–	codifiable

The closure state has $3 \times 3 \times 4 = 36$ possibilities. For the specified closure mechanism, the state 223 has the highest "closure degree" and corresponds to a "closed" arc or fact. The particular closure's state is a *subjective* appreciation for the agent, in the sense that it does not tell anything to an observer who does not have precise knowledge of the mechanism.

3.2 Behavior Modulators

We are interested in the emotional modulation of cognition [3, 4]. In this sense, emotions are the observable result of a particular set of values for the behavior modulators. We will consider a humble approach to Dörner's theory [4] to test the idea that modifying the behavior according to the actual knowledge state, *closure state* of an agent can help obtain more knowledge. We use a single behavior modulator, called *focus*. This parameter, with values between 0 and 1, modulates the probability to revisit the previous sensing state, by undoing the last performed movements. It is called focus because it is a mechanism used to perceive again something by trying (with a probabilistic measure) to revisit a situation. A focus with value 0.0 means that the agent will always move in a random direction. A focus with value 1.0 means that the agent will always undo the last movement. The different focus values can be interpreted by an observer as different emotional states. For example a high focus value could be seen as "interest".

4 Experiments with Internal Representation

The experiments will be described from two perspectives, one internal corresponding to the closure mechanism's dynamics trying to incorporate signals and actuations, and other external corresponding to a network's quantifiers as being macro variables. We are interested in the way the focus affects the representation's development given the system and the agent's closure mechanism. Every time the system changes closure state, either by incorporating a node, an arc, or changing their status in the representation, the agent has integrated more knowledge from the environment and its interrelationship, captured in the network.

4.1 Closure Dynamics with Constant Focus

In a first set of experiments, we perform runs of the feeding game, varying the focus value. All agent's movements are random, but when focus value > 0, the last movement can be undone. Closure dynamics or knowledge incorporation process builds a probabilistic network, where closure states/nodes and transitions between them (arcs) are weighted by the ocurrence frequency.

Two types of arcs are recognized: loops and simple-transitions. Loops represent time without knowledge acquisition, Table 4.1 shows some loops' relative frequencies for each focus. As the last row of the table shows, the agent is engaged in loops in more than the 85% of the total time. During loops, there is no distinguishable change in the closure mechanism. The number of loops changes according to the focus value; a lower loop frequency indicates less time without changes in the representation, learning faster. The focus plays a different role in learning depending on the specific loop. There is no "best" focus value, but the "optimal" value depends on the actual (internal) context.

Table 2. Relative frequencies of loops (=0.01)

	Focus value				
loops	**0**	**0.25**	**0.5**	**0.75**	**var**
222-222	0.31	0.26	0.25	0.23	0.17
000-000	0.12	0.15	0.16	0.24	0.09
223-223	0.09	0.09	0.12	0.14	0.19
221-221	0.08	0.06	0.07	0.08	0.14
121-121	0.05	0.06	0.06	0.05	0.09
111-111	0.05	0.05	0.04	0.03	0.01
211-211	0.05	0.05	0.05	0.04	0.07
100-100	0.04	0.05	0.05	0.04	0.03
010-010	0.04	0.05	0.05	0.04	0.02
200-200	0.03	0.02	0.02	0.01	0.02
total	0.86	0.85	0.87	0.91	0.83

When the system experiences a change in the closure state, we can say that the system has incorporated structure in the representation. Table 4.1 shows the relative frequencies. The total time the agent develops its representation (transitions) is lower than the time devoted to loops.

Table 3. Relative frequencies of transitions (=0.01)

	Focus value				
transitions	**0**	**0.25**	**0.5**	**0.75**	**var**
110-111	0.05	0.05	0.04	0.01	0.03
020-021	0.03	0.03	0.02	0.01	0.03
210-211	0.02	0.03	0.03	0.02	0.04
120-121	0.01	0.01	0.02	0.01	0.02
221-223	0.01	0.01	0.01	0.01	0.02
220-221	0.01	0.01	0.01	0.01	0.02
121-223	0.00	0.00	0.00	0.00	0.01
total	0.14	0.15	0.87	0.91	0.83

4.2 Taking Advantage of the Closure's Dynamics: Variable Focus

After analyzing the probabilistic networks corresponding to the closure structures obtained with each of the focus values. We repeated the feed game experiment but changing the focus value in function of the actual closure state according to the following rule: If $closureState \in \{222, 221, 211, 121, 212, 223, 213\}$ then *focus* = 0.66 else *focus* = 0.0. The system changes its focus value in a reactive way, depending only on the current closure state, not on the sensing states. The goal is not to find the "best" focus value for each closure state, but just to show that modifying the behavior modulator in terms of the closure state affects the acquired knowledge, obtaining a different structure of the global representation. The results are shown in the last column of Table 4.1 and Table 4.1. We can see that the loops in which the agent spends more time decrease sensibly, and that the transition frequency rise or are maintained except for the 110-111 case. A detailed analysis of the data shows that the paths to the state 223 (facts) are favored.

4.3 External Description: Network Properties

Both loops and transitions as considered before are characteristic of the closure mechanism, reflecting the dynamics during the development of the network. We can calculate the closure state distribution obtained from the resulting representation considering all the existing arcs and their associated nodes, as shown in Table 4.3. This distribution can be considered as an external observation, because it is a "picture" of the representation at a certain time, but does not give information on how the arcs have obtained their closure state. We can observe again that the focus affects the closure state of arcs in different ways. For facts, there is no significant variation with fixed focus values. But their frequency is increased in an important way ($\approx$ 50%) with variable focus.

Table 4. Closure state of arcs of final representation

	Focus				
arcs	**0**	**0.25**	**0.5**	**0.75**	**var**
121	586	539	503	243	586
221	386	338	271	272	409
211	264	341	364	191	447
Fact: 223	213	188	204	211	307
111	370	477	337	97	196
222	59	59	43	43	94
213	6	1	8	4	6
212	1	1	2	0	0
num arcs	1885	1944	1732	1061	2044

We used the Clustering Coefficient [17] of the representation network as a measure of global structure. The clustering coefficient shows that the more efficient and more stable case is the one of variable focus. The variable behavior modulator actuates internally to produce improvements in the external structure. In our model, the sensed turns into signal internally through the closure process, and the result can be measured externally in the structural properties of the representation network. We also analyzed the subnets related to each biological motivation (data not shown). Each subnet is similar to a scheme, more in the Piagetian sense than in Drescher's sense. This is because the subnet corresponds to structured knowledge with some biological meaning and not to concrete context - actuation-result detection. The structural properties of subnets are better with a variable focus than with a fixed one. There are also more subnets with a variable focus, giving the possibility to develop more schemes.

5 Conclusions and Future Work

In this work, we try to isolate a representation's evolving process, avoiding any use of "cognition" about the "state of the world", to explore two aspects: 1) A case study on how the internal and external modes of description can be related, and; 2) How behavior modifiers based on internal considerations can affect the structural properties of the developed representation. The closure mechanism is in itself a *knowledge acquisition mechanism* in the sense that it incorporates in the representation the structural relationships with the environment. Using a behavior modulator called focus, the representation and its structure can develop in different ways. A selective value for specific closure states improves the structural properties of the representation (external mode of description).

Our model is not a behavior-based nor knowledge based. It is atypical for a "knowledge acquisition mechanism", since the agent does not react directly to its world. However, the focus can modify the behavior patterns. The variable focus allows the agent to react to its knowledge state in order to incorporate faster the relationships with its environment. The obtained representation does not catch *structural* properties of the environment, but makes explicit the structural interactions between the agent and environment. We have avoided any use of the representation but these have a potentiality to be used. Only structural or historical embodiment is not enough for obtaining autonomously rich representations. It seems the same as to think about only affordances. We need consider also an internal process *independent to the world's dynamics,* in such a way that the representation becomes richer. Note that the mentioned dynamics is different from the related with the use of representations. In the agent's life, both dynamics are crucial, and must be related, but at this moment we are concentrated in building representations. Our work can be seen as a kind of learning representation without explicit goals, i.e., an implicit learning process.

As a future work, we can see several directions which could be followed. Intersubjective representations could be obtained by pragmatic games in which two or more agents interact with an environment. This topic is interesting for studies in the evolution of communication. Another direction would be to study the effect of different behavior modifiers in the development of the representation.

Acknowledgements

C. R. de la M. B. was supported by CONACyT, México and Universidad Veracruzana. C. G. was partially supported by CONACyT, [[México]].

References

1. Brooks, R.: Elephants Don't Play Chess. Robotics and Autonomous Systems 6: 3-15 (1990)

2. Collier, J.: Autonomy and Process Closure as the Basis for Functionality. Chandler, J.L.R and Van de Vijveir, G. (eds.) Closure: Emergent Organizations and their Dynamics. Annals of the New York Academy of Science 901: (2000)
3. Dolan, R. J.: Emotion, Cognition, and Behavior Science 298: 1191-1194 (2002)
4. Dörner, D.: The Mathematics of Emotions. Fifth International Conference on Cognitive Modeling, ICCM Bamberg, Germany, April 10-12, (2003)
5. Drescher, D. L.: Made-Up Minds: A Constructivist Approach to Artificial Intelligence. MIT Press, (1991)
6. Hillman, C.: A Categorical Primer. (1997) Unpublished.
7. Maturana, H. and Varela, F. The Tree of Knowledge. Shambala Press. Boston, MA (1987).
8. Mitchell, M.: Theories of Structure Versus Theories of Change. Behavioral and Brain Sciences 21(5) (1998)
9. Montangero, J., Maurice-Naville, D.: Piaget or the Advance of Knowledge. Lawrence Erlbaum Associates, Publishers. New Jersey, London, (1997)
10. Rocha, Luis M.: Evolution with material symbol systems. Biosystems 60:95-121 (2001)
11. Scheutz, M., Sloman, A.: Affect and agent control: experiments with simple affective states. Proceedings of IAT-01, World Scientist Publisher (2001)
12. Sipper, M., Sanchez, E., Mange, D., Tomassini, M., Prez-Uribe, A., Stauffer, A.: A Phylogenetic, Ontogenetic, and Epigenetic View of Bio-Inspired Hardware Systems. IEEE Transactions on Evolutionary Computation 1, April (1997)
13. Steels, L.: Self-organizing vocabularies. Langton, C. (ed.) Proceedings of Artificial Life V. Nara, (1996)
14. Steels, L.: Intelligence with representation Phil. Trans. Real Soc. Lond. A. 361: 2381-2395 (2003)
15. Strogatz, Steven: Exploring complex networks. NATURE, vol 410. March 2001 pp 268-276. (2001)
16. von Uexkll: J. Theoretische Biologie. Springer Verlag, Berlin, Germany (1928)
17. Watts, J. and Strogatz, S.H.: Collective dynamics of ‘small-world’ networks. Nature Vol. 393: 440-442. (1998)
18. Zlatev, J., and Balkenius, C.: Introduction: Why “epigenetic robotics”?. Proceedings of the First International Workshop on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems. Lund University Cognitive Studies, Volume 85: 1-4 (2001)
19. Ziemke, T.: Are Robots Embodied?. First International Workshop of Epigenetic Robotic: Modeling Cognitive Development in Robotic Systems. September 17-18, 2001. Lund, Sweden. (2001)

New Technique to Improve Probabilistic Roadmap Methods

Antonio Benitez and Daniel Vallejo

Department of Computer Engineering,
Universidad de las Américas Puebla,
CP 72820, México
{sc098381, dvallejo}@mail.udlap.mx
http://www.udlap.mx/sc098381

Abstract. A probabilistic roadmap is a network of simple paths connecting collision-free configurations obtained by sampling a robot's configuration space at random. Several probabilistic roadmap planners have solved unusually difficult path planning problems, but their efficiency remains disappointing when the free space contains narrow passages. This paper provides a new technique to find free configurations into narrow corridors, sampling the configuration space using geometric features into the workspace and computing configurations close to the obstacles. An initial roadmap is built using spheres in low cost, next an improving connectivity phase based on *"straightness", "volume" and "normal vectors"* features on the workspace is computed, and the roadmap is improved capturing a better connectivity of configuration space. Experiments show that the new approach is able to solve different benchmarks in motion planning problems containing difficult narrow corridors.

Keywords: Probabilistic roadmap methods, geometric characteristics, robotics.

1 Introduction

Motion planning in the presence of obstacles is an important problem in robotics with applications in other areas, such as simulation and computer aided design. While complete motion planning algorithms do exist, they are rarely used in practice since they are computational infeasible in all but the simplest cases. For this reason probabilistic methods has been developed. In particular, several algorithms, known collectively as *probabilistic roadmap planners,* have been shown to perform well in a number of practical situations, see, e.g.,[9]. The idea behind these methods is to create a graph of randomly generated collision-free configurations with connections between these nodes made by a simple and fast local planning method. These methods run quickly are easy to implement; unfortunately there are simple situations in which they perform poorly, in particular situations in which paths are required to pass through narrow passages in configuration space [11].

The geometry in workspace into motion planning problems has been used to propose several heuristic based in medial axis or generalized Voronoi diagrams, see [4, 10]. In particular, in two dimensions the medial axis is a one dimensional graph-like structure which can be used as a roadmap. However, the medial axis is difficult and expensive to compute explicitly, particularly in higher dimensions.

1.1 Our Results

We propose a new algorithm which combines these two approaches by generating random networks whose nodes lie on the obstacles surface. Our central observation is that it is possible improve the connectivity of configuration space using geometric features on the workspace to find free configurations close to the obstacles. The main novelty in our approach is a new method for generating roadmap candidate points. In particular, we attempt to generate candidate points distributed close to each obstacle on work-space taking advantage on their geometric features. Using this approach, high quality roadmaps can be obtained even when work-space is crowded. Experimental results with *"free flying objects"* with six degrees of freedom (dof) will be shown.

Previous results using the new proposal have been presented in [5], and in this paper we have improve the heuristic adding a new feature, called *"normal vector"*.

The approach extends fairly easily to dynamic environments. Our approach can be applied to some important situations that have so far not been satisfactorily solved by heuristic methods (Paths through long, narrow passages in crowded Work-space can be found).

The previous approaches related to ours are the path planning methods of Kavraki and Latombe [8], Overmars and Svestka [12,13] mentioned above. In fact, in [13] the authors describe a technique they call geometric node adding in which roadmap nodes are generated from robot configurations near obstacle boundaries, which is very similar to the idea of generating nodes on C-obstacle boundaries [3].

We describe how the roadmap is constructed in Section 2, and how it is used for planning in Section 3. Implementation details and experimental results are presented in Section 4 and 5, respectively.

1.2 Probabilistic Roadmap Methods

Probabilistic roadmap methods generally operate as follows, see, e.g.,[9]. During a preprocessing phase, a set of configurations in the free space is generated by sampling configurations at random and removing those that put the workpiece in collision with an obstacle. These nodes are then connected into a roadmap graph by inserting edges between configurations if they can be connected by a simple and fast local planning method, e.g., a straight line planner. This roadmap can then be queried by connecting given start and goal configurations to nodes in the roadmap (again using the local planner) and then searching for a path in the roadmap connecting these nodes. Various sampling schemes and local planners have been used, see [2, 7, 13]. The algorithms are easy to implement, run quickly, and are applicable to a wide variety of robots.

The main shortcoming of these methods is their poor performance on problems requiring paths that pass through narrow passages in the free space (C_{free}). This is a direct consequence of how the nodes are sampled. For example, using the usual uniform sampling over C*–space,* any corridor of sufficiently small volume is unlikely to contain any sampled nodes whatsoever. Some effort has been made to modify sampling to increase the number of nodes sampled in narrow corridors. Intuitively, such narrow corridors may be characterized by their large surface area to volume ratio: the method in [2] and [6] have exploited this idea.

2 Roadmap Construction

The task required for building the roadmap are generating the roadmap candidate nodes and connecting the candidates to form the roadmap. The description of these tasks below is for a free-flying rigid body in three dimensions.

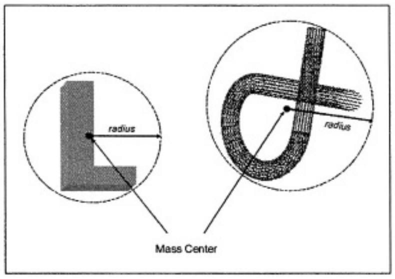


Fig. 1. This figure presents the geometric center, in the first sample the geometric center is placed into the body, in the second one this metric is computed out of the mesh

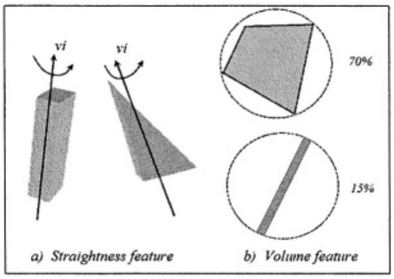


Fig. 2. a)The rotation axis is defined in the same direction of the *straightness* feature, b) The *volume* feature is proportional to the sphere volume

2.1 Algorithm Description

The description of the algorithm is divided into four parts: geometric features to be used, first approximation of the configuration space, improving the connectivity and planning. The following sections describe how the algorithm works and we give some details about its implementation. Some figures are included to show the main idea behind this new proposal.

2.2 Geometric Features

The workspace is read from different files, these files contain the triangle meshes which represent the geometry of each body into the workspace. Using such representation, the algorithm compute several parameters which will be used by the heuristic and are defined as following:

Mass Center. This metric is calculated as the average of the x_s, y_s, and z_s values of the vertex for a given body into the environment. This parameter might not be placed into the object. This property can be see in figure 1.

The Body Radius. This parameter is computed using the distance between the center mass and the farest vertex into the object. This metric is used to calculate the sphere which the object will be surrounded. The figure 1. shows the position where the mass center is placed and the radius computed using this metric.

Straightness. Let q_i be the vector which define the direction of the *"straightness"* feature for each $V_i \in B$, and qr_i will define the same feature on the robot. This feature indicates the direction which the body presents its long side. The figure 2 a), shows how this feature can be see for a given object.

Volume. Let vol_i be the *"volume"* of the obstacle V_i with respect the volume of the sphere used to surround it. Both, the *"straightness" and "volume"* features are used to improve the connectivity of the roadmap and the figure 2 b). shows a geometric representation of this parameter.

Normal Vector. The normal vector, often simply called the "normal," to a surface is a *vector perpendicular* to it. Often, the normal *unit vector* is desired, which is sometimes known as the "unit normal."

2.3 First Sampling of C-Space

During this stage, the algorithm uses spheres to surround the robot and the obstacles. The figure 4. shows the view of the first sampling. Using spheres only during this stage we have two advantages, first the robot has the characteristic to rotate in any direction, which will be used for the local planner in the improving stage, and second, the cost of collision detection is reduced, because the routine is limited to detect when two spheres are in collision (the sphere associated to each obstacle and the other one associated to the robot). In this process just a small number of configurations will be computed. The main idea in this stage is to place configurations in open space and to spend few time. Perhaps the technique lose some free configurations but this proposal is focus to compute difficult configurations, therefore these lose configuration can be calculated in the second stage.

2.4 Improving the Connectivity of C-Space

If the number of nodes computed during the first approximation of the roadmap is large enough, the set *N* gives a fairly uniform covering of C_{free}. In easy scenes *R* is well connected. But in more constrained ones where C_{free} is actually connected, *R* often consists of a few large components and several small ones. It therefore does not effectively capture the connectivity of C_{free}.

The purpose of the expansion is to add more nodes in a way that will facilitate the formation of the large components comprising as many of the nodes as possible and will also help cover the more difficult narrow parts of C_{free}.

In this phase, we generate a set *N* of candidate roadmap nodes, each of which corresponds to a point in C-space. The general strategy of the node generation

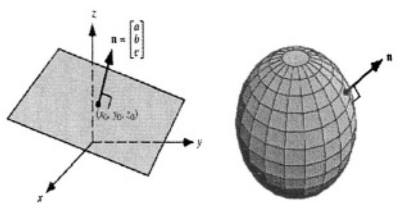


Fig. 3. This figure shows the geometric representation of the normal vector

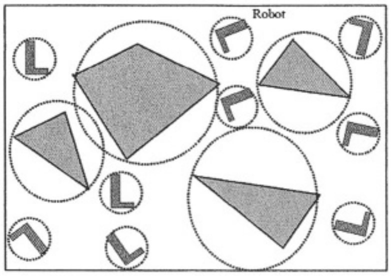


Fig. 4. First sample of configurations space using spheres to sorround the objects and reduce the cost of collision detection during this stage

process is to construct a set N_i of candidate nodes for each object V_i such that each $c(V_i) \in N_i$ lies near to obstacle V_i. The set of roadmap candidate nodes is the union of the candidate sets computed for each obstacle $V_i \in B$, i.e., $N = \cup_i N_i$.

Node Generation. We now consider how to compute the candidate set for each obstacle. During this stage the technique attempt to take advantage of some geometric features of the robot and the obstacles to obtain information that allows guide the search of useful configurations.

Paralell and Perpendicular Configurations. The first geometric feature used is called *"Straightness"*. This feature indicates the direction which the object presents its long side and it is given by a vector v_i, which we have used as the direction of the rotation axis. The Figure 2 a). shows the way which the rotation axis is represented on the robot and the obstacles. We can see that the object will sweep a small volumen as result of defining the rotation axis in the same direction of the straightness feature.

The main idea behind the parallel and perpendicular configurations is to compute configurations close to the obstacles, and to place the robot in that way that, when the algorithm attempts to rotate the robot (searching a free configuration) using its rotation axis, the volume swept will be small avoiding get in collision.

Now we are going to describe how to generate m points close to the obstacle V_i. Using the *straightness* feature, the next algorithm compute parallel and perpendicular configurations.

1. First, the algorithm search a collision configuration $c(V_i)$ on the V_i obstacle, such $c(V_i)$ is calculated uniformly distributed around the sphere which the obstacle is surrounded (a vicinity for each V_i isdefined).

2. Next, the technique attempts to rotate this configuration until it will be parallel to the obstacle (that is $v_{ri} \parallel v_i$), if the parallel configuration is not incollision then it is added to N_i, else the process called *elastic band* is applied searching to turn it in free configuration, which will lie close to the obstacle V_i.

3. Once a parallel configuration has been processed, the algorithm computes the perpendicular configuration (that means that $v_{ri} \perp v_i$) taking the $c(V_i)$ calculated in step 1. In the same way like in step 2, if the new perpendicular configuration is not in collision then it is added to N_i, else the *elastic band* process is applied. The figure 5, shows how the parallel and perpendicular configurations can be seen around the V_i.

The second strategy to calculate free configurations close to the obstalces is to place the robot parallel to the direction of a normal vector of some triangle on the obstacle surface.

So, the heuristic calculates the geometric center for each triangle in the mesh and obtains the normal vector for each triangle (this process is included as a preprocessing phase and is computed for each body in the environment).

The idea behind this process is to compute free configurations close to the ostacle surfaces. After the process have calculated parallel and perpendiacular configurations, the heristic attempt to obtain random configuration using as direction of rotation axis the direction of the normal vector.

The Elastic Band Algorithm. This process works as following, first the heuristic calculates the distance vector d_i between the obstacle position and the configuration $c(V_i)$, and attempt to approach and moving away the robot with respect to the obstacle. To compute this operation, the process scale the d_i vector (using the values computed with respect the *"volume"* feature) to calculate the next position where the $c(V_i)$ will be placed.

The metric used to obtain the distance vector is the Euclidean distance in three dimensional space, and the scale factor is computed using the *"volume"* feature of the objects. Thus, the scale factor will be able to be initialized since a low value (MIN), these values are presented in figure 7., and for each interaction the vector will increase until reach the maxima value (MAX). This

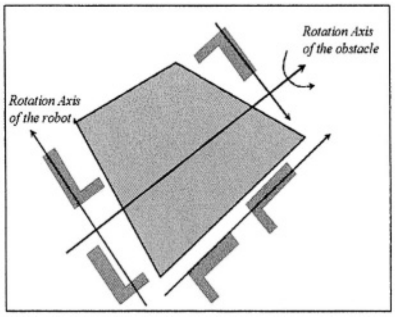


Fig. 5. Parallel and perpendicular configurations computed close to the object

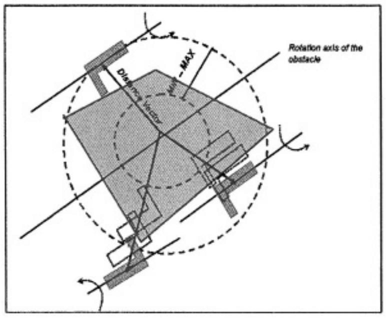
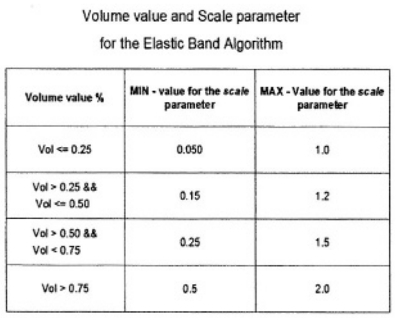


Fig. 6. Elastic band process, approaching and moving away the configuration from the obstacle using a scalar metric on the distance vector d_i



Volume value and Scale parameter
for the Elastic Band Algorithm

Volume value %	MIN - value for the scale parameter	MAX - Value for the scale parameter
Vol <= 0.25	0.050	1.0
Vol > 0.25 && Vol <= 0.50	0.15	1.2
Vol > 0.50 && Vol < 0.75	0.25	1.5
Vol > 0.75	0.5	2.0

Fig. 7. This table presents the values for MAX and MIN parameters used during the elastic band process

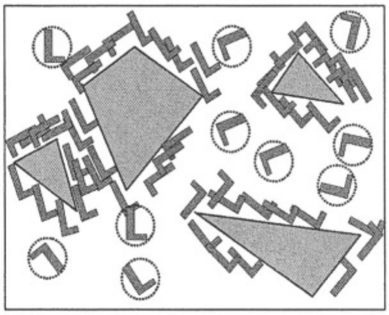


Fig. 8. This figure presents the final connectivity of configuration space (after to apply the Elastic Band process)

metric has an important role, because we are interesting in configurations close to the objects, that means that, the mass center of the objects will have to be near.

The elastic band process works with parallel and perpendicular configurations which are computed close to the obstacle. While the distance vector is computing the next configuration to be tested, the robot is rotated on its rotation axis, searching to find a free configuration (taking advantages of the straightness feature). The figure 6. shows how the parallel and perpendicular configurations are computed near to the obstacle and how the scalar vector is changing, approaching and moving the robot away from the obstacle. The configurations which are marked with dots are calculated during the process until reach a free one.

Once the improving strategy has been applied, the connectivity of the roadmap could be see as in figure 8. the figure presents the configurations calculated during the first approximation (which are surrounded by a sphere), and we can see how the heuristic is able to place many configurations in narrow regions to reach a better connectivity of configuration space.

2.5 Connecting Roadmap Candidates

We now consider how to connect the candidate nodes $N = \cup_i N_i$ to create the roadmap. The basic idea is to use a simple, fast, local planner to connect pairs of roadmap candidate nodes.

Ideally, the roadmap will include paths through all corridors in C-space. Thus, a trade-off exist between the quality of the resulting roadmap and the resources (computation and space) one is willing to invest in building it.

Many different connection strategies could be used in path planning applications. In the prevous work [5] we only use the method used in [8] trying to connect each node $c(V_i) \in N$ to its k nearest neighbors in N.

3 Planning

Planning is carried out as in any roadmap method: we attempt to connect the nodes x_1 and x_2, representing the *start* and *goal* configurations, respectively, to the same connected component of the roadmap, and then find a path in the roadmap between these two connection points. The following approach, proposed in [8], is well suited for our roadmap.

If no connections is made for x_i, then we execute a *random walk* and try to connect the initial or the end node to the roadmap. This can be repeated a few times if necessary. If we still can not to connect both nodes to the same connected component of the roadmap, then we declare failure. After both connections are made, we find a path in the roadmap between the two connection points using Dijkstra's algorithm. Recall that we must regenerate the path between adjacent roadmap nodes since they are not stored with the roadmap.

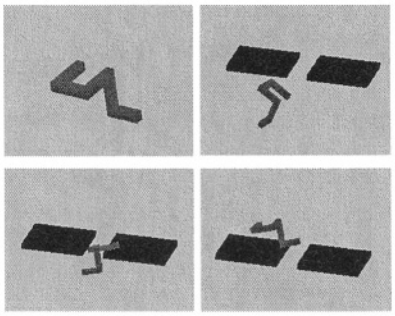


Fig. 9. The form which the robot is presented is more complex

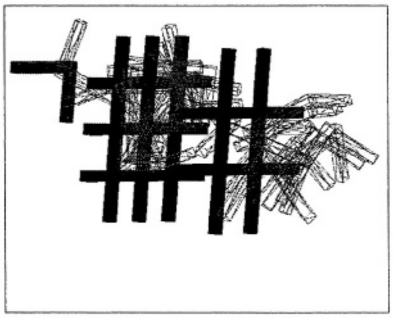


Fig. 10. This sample showsn two grids and the robot has to pass though them

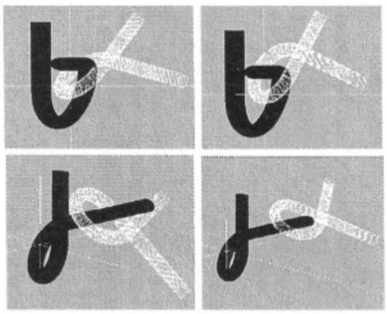
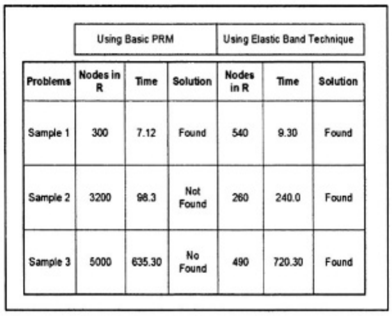


Fig. 11. The *alpha puzzle* problem version 1.2



	Using Basic PRM			Using Elastic Band Technique		
Problems	Nodes in R	Time	Solution	Nodes in R	Time	Solution
Sample 1	300	7.12	Found	540	9.30	Found
Sample 2	3200	98.3	Not Found	260	240.0	Found
Sample 3	5000	635.30	No Found	490	720.30	Found

Fig. 12. The information in the table is the average after ten runnings for each sample. The time is showed in minutes

4 Experimental Results

We implemented a path planner for free flying objects with six degrees of freedom in a three dimensional workspace. The code was written in C++ on PC Intel Pentium 4, the CPU was a 2.4 Ghz with 512MB of RAM.

In the following, we analyze the performance of the method (this performance is seen since the capability of the method to solve the problems) on few scenes. In all cases we used a free-flying object robot with six dof. The various environments, and some representative configurations of the robot, are shown in Figures 9,10 and 11. The three samples shown are presented as result of the technique applied on the problems. We present three problems, they have different difficult level. The problems are labeled as Sample1, Sample2 and Sample3. Below we discuss the environments in more detail.

Sample1: This scene is presented with two obstacles and we can see that the form of the robot is more complex. There is a narrow corridor which becomes difficult to solve, nevertheless, the heuristic is able to find a path which goes through the corridor. The figure 9. presents how the robot pass through the corridor.

Sample2: This environment is presented with two grids and the form of the robot is like a "L" . There are several narrow passages, and the distance between the grids is small, therefore the robot has a reduced area to move and rotate between the grids. The figure 10. shows the how the heuristic was able to find a path between the start and goal configurations.

Sample3: This problem is well know as the *"alpha puzzle"* problem. There exist several different versions of this problem [1]. The Figure 11. shows the solution for the 1.2 version of the problem. This problem has the difficult of having few configurations into the corridor, therefore its solutions is very complicated. Our method is able to find a path to solve it. In the figure some configurations into the corridor are shown.

Table in figure 12. shows the results of performance of the Basic PRM algorithm in comparison with the Geometric Features based PRM algorithm proposed in this work. We can see that the time used by the new proposal in larger that the used by the basic PRM, nevertheless we think that this time is well used (because the a solution can be found).

5 Conclusion

We have described a new randomized roadmap method for motion planning problems. To test the concept, we implemented the method for path planning for *"free flying object"* in a three-dimensional space. The method was shown to perform well. Currently, and we can say that geometric features on the workspace can be used to built heuristics to guide the search of free configurations into narrow corridors. We keep on working on the free flying objects problems, and we are geometric features on the heuristic searching to improve the connectivity of configuration space.

References

1. N. Amato. Motion Planning Puzzels Benchmarks, http://parasol.tamu.edu/ amato/
2. N. Amato, B. Bayazit, L. Dale, C. Jones, and D. Vallejo. Obprm: An obstacle-based prm for 3D workspaces. *In P.K. Agarwal, L. Kavraki, and M. Mason, editors, Robotics: The Algorithm Perspective.* AK Peters, 1998.
3. N. M. Amato and Y. Wu. A randomized roadmap method for path and manipulation palnning. *In Proc. IEEE Internat. Conf. Robot. Autoum,.* Pages 113-120, Mineapolis, MN, April 1996.
4. F. Aurenhammer. Voronoi diagrams: Asurvey of a fundamental geometric data structure. *ACM Comp. Surv.*,23:345-405,1991.
5. A. Benitez, D. Vallejo, and M.A. Medina. *PRMs Based on Obstacles' Geometry.* In Proc. IEEE The 8th Conference on Intelligent Autonomous Systems (IAS-8), Amsterdam, The Netherlands 10-13 March 2004. Pag. 592 - 599.
6. D. Hsu, L. E. Kavraki, J. C. Latombe, R. Motwani, and S. Sorkin. On finding narrow passages whit probabilistic roadmap planners. In *Proc. 1998 Workshop Algorithmc Found. Robot.,* Weslley, MA, 1998. A.K. Peters.
7. L. Kavraki. *Random Networks in Configuration Space for Fast Path Planning* PhD thesis, Stanford Univ., Stanford, CA, 1995.
8. L. Kavraki and J.-C. Latombe. Randomized preprocessing of configuration space for fast path planning. *In Proc. IEEE Internat. Conf. Robotics and Automation.,* pages 2138-2145, San Diego, CA, 1994.
9. L. Kavraki, P. Svestka, J.C. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *In Proc. IEEE Internat. Conf. Robot. Autoum,.* 12(4): 566-580, August 1996.
10. J-C. Latombe. Robot Motion Planning. Kluwer Academic Publishers, Boston, MA, 1991.
11. T. Lozano-Pérez. Spatial planning: a configuration space approach. *IEEE Tr. On Computers,* 32:108-120, 1983.
12. M. Overmars. A Random Approach to Motion Planning , *Tecnical Report RUU-CS-92-32,* Computer Science, Utrecht University, the Netherlands, 1992.
13. M. Overmars and P. Svestka. A probabilistic learning approach to motion planning. *In Proc. Workshop on Algorithmic Foundations of Robotics.,* pages 19-37 1994.

A New Neural Architecture Based on ART and AVITE Models for Anticipatory Sensory-Motor Coordination in Robotics

J.L. Pedreño-Molina[1], O. Flórez-Giráldez[2], and J. López-Coronado[2]

[1] Department of Information Technologies and Communications
[2] Department of Systems Engineering and Automation, Campus Muralla del Mar, s/n Technical University of Cartagena, 30.202 Cartagena, Murcia, Spain
{Juan.PMolina, Oscar.Florez, Jl.Coronado)@upct.es

Abstract In this paper a novel sensory-motor neural controller applied to robotic systems for reaching and tracking targets is proposed. It is based on how the human system projects the sensorial stimulus over the motor joints, sending motor commands to each articulation and avoiding, in most phases of the movement, the feedback of the visual information. In this way, the proposed neural architecture autonomously generates a learning cells structure based on the adaptive resonance theory, together with a neural mapping of the sensory-motor coordinate systems in each cell of the arm workspace. It permits a fast open-loop control based on propioceptive information of a robot and a precise grasping position in each cell by mapping 3D spatial positions over redundant joints. The proposed architecture has been trained, implemented and tested in a visuo-motor robotic platform. Robustness, precision and velocity characteristics have been validated.

1 Introduction

One of the topics in robotics is the problem of solving the inverse kinematics of redundant visuo-motor systems for reaching applications in real time. Most of the proposed solutions are based on close-loop control systems. They are highly dependent on the vision system and also need to track the entire robot arm end-effector trajectory. Although these control systems are continuously employed in robotic and good results are obtained [1], the sensory-motor coordination human system does not require the visual tracking of the joints whose propioceptive information is learning [2] during action-perception cycles, mainly during the child phase.

The main objective of this work is to give a solution for solving the inverse kinematics of robots, without the knowledge of the internal physical properties of the robot arm, such as joint lengths and rotation and translation thresholds of each joints. One algorithm giving that solution has the advantage of avoiding the continuous calibration of the system and simultaneously to be independent from the considered robotic platform. The information propiocetive will need to be learned by mapping the end-effector 3D spatial position, given by the vision system, and the joint positions configuration, given directly by the motor encoders.

One of the difficulties in this work was the necessity to have a totally well-mapped spatial-motor and motor-spatial information [3], using previous learnt information for anticipatory planning an action program. In this way, the actions are produced quickly without a close-loop. The final workspace of the robot arm is autonomously divided in small structures like learning cells. The proposed model aims at the idea of solving the accuracy sensory-motor coordination by means of two neural networks whose interconnection allows the anticipatory behaviour of the model. This interconnection is based on the self-organizing Adaptive Resonance Theory (ART algorithm) for discrete processes [4], and the AVITE model (Adaptive Vector Integration to End Point) [5]. The ART algorithm is a self-organizing neural network which has the ability of solving the stability–plasticity dilemma for the competitive learning phase. Uses of this algorithm to the proposed architecture will permit to carry out the described anticipatory behaviour. In the other hand the AVITE neural model, based on supervised learning, permits to map the spatial-motor positions in each learning cells. As results, the proposed work is capable to combine visual, spatial, and motor information for reaching objects by using a robot arm, tracking a trajectory in which the close-loop control is only carried out in each learning cell of the workspace. The proposed architecture has been implemented in an industrial robot arm and capabilities of robustness, adaptability, speedy, accuracy have been demonstrated for reaching tasks, including perturbations in the objective position.

2 Neural Model Structure: Self-Organizing and Fast Mapping

The proposed architecture is based on two interconnected neural models that sequentially project the 3D final position (sensorial information) of the object to be grasped over the joint positions (spatial information) of the robot arm end-effector. This task is made in a predictive way by means of adaptive distribution of the workspace. The base of the control scheme is to generate random movements of the robot arm, whose end-effector position is detected and computed by a vision system and then the robot arm 3D workspace is divided into small cells in whose centres the precise position of the robot joints are well known, by means of the propioceptive information and one previous learning phase. It produces, in the operation phase, an anticipatory movement of the robot toward the centre of the cell in which the target is located. The supervised neural model based on the ART algorithm includes a vigilance parameter controlling the competitive learning and the final position and dimension of each cell. By means of a second learning phase, one neural weight map is obtained for each cell. Due to the lineal nature of the AVITE model, the spatial-motor projection is quickly computed and few steps close-loop control are required for accurate reaching tasks. The AVITE model projects the difference vector *(DV),* in visual coordinates between the current and desired position of the end-effector, over the incremental angular positions of the robot arm. Thus, the 3D visual distance inside the winner cell is reduced with high precision and fast operation. The general performance of the proposed model is represented in the scheme of the Fig. 1.

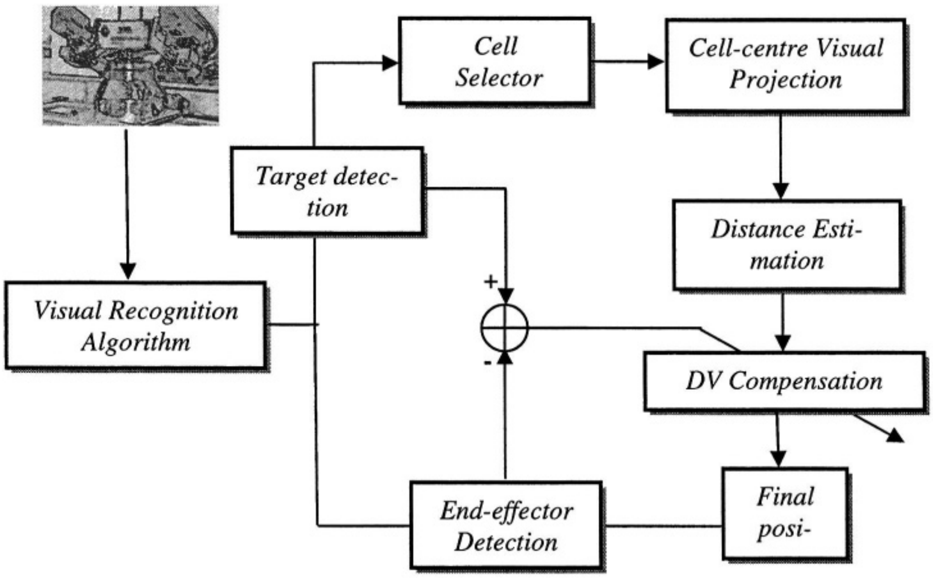


Fig. 1. General scheme of the neurocontroller. It is formed by two interconnected neural models to map the 3D workspace (non-supervised model) and to compensate the spatial error between the current and desired final spatial position (supervised AVITE model)

In this neural model, the vision system of the stereohead detects the position of the object to be grasped. The internal representation of that position will be the input to the cell selector module. By means of a competitive algorithm, this module calculates the cell in whose workspace is located the target. The projection of the visual position of the centre of the cell over the arm joint positions is achieved by the *cell-centre visual projection module.* Once the AVITE model has been executed, the difference between the centre of the cell and the desired position, in visual coordinates *(DV),* is estimated by means of the *distance estimator module.* Then, the *DV compensation* module reduces that distance by means of few robot arm movements and lineal projections. Finally, the produced error is used to update the neuron weights of the AVITE model. It will permit to detect if an unexpected situation happens or if a mechanical blocking in some joints of the robot arm is produced. In order to validate the behaviour of the proposed architecture operating in dynamic environments, perturbations to the target position have been considered and the performance of cells commutation when tracking moving objects is tested. The obtained results emphasize the emulations of human biological behaviour for the proposed architecture. In a human system the majority of the time for reaching objects is dedicated to the movement compensation due to possible perturbations in the measurements of both sensors: visual and propioceptive. The associative maps which are generated by the AVITE algorithm, permits to learn the gesture of the robots, including the mechanical faults of the robot system.

3 Non-supervised Adaptive Generation of Learning Cells

The non-supervised neural model implemented in the proposed architecture is based on the ART2 model developed by Carpenter, et al. [4]. It is focused to the workspace division in spatial coordinates and to supply the anticipatory behaviour to the neural architecture. Each 3D region will be different and characterized by the position of its centroids and the Voronoi frontiers, implying the configuration of the final dimension of each cell. How the cellular structure is defined in the spatial frame, it will determine the number of steps and the precision or final error of the neural model for reaching tasks. The structure of the ART model allows to control the final cell configuration by means of one vigilance parameter and the learning trial number. The structure of the ART neural network is represented in Fig.2.

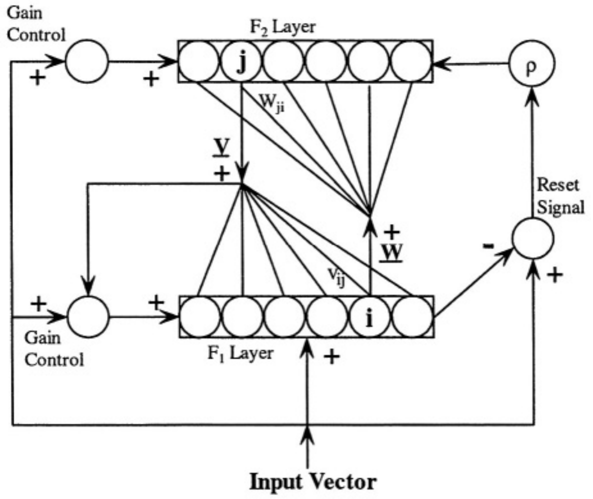


Fig. 2. ART structure for adaptive generation of learning cells. The input layer F_1 has the same dimension like the input vector; The neurons (centroids) of the output layer F_2 are the patterns to be classified; W_{ji} are the feed-forward connection weights; V_{ij} are the feedback connection weights; The *Gain Control* is used to get network stability by inhibition of activation control of the F_1 and F_2 layers; The *Reset Signal* is used to control the membership level of a pattern to the winner neuron in F_2 layer and, finally, ρ is the vigilance parameter

In the learning phase, initially one random posture for the robot arm is generated and both end-effector spatial positions and target, referred to the robot arm coordinate frame, are computed by the vision system. Taking *D* the number of d.o.f. of the robot, each position is represented by the θ_v vector (*1xD* dimension) while its corresponding end-effector position is represented by P_{xyz} vector (*1x3* dimension). A ρ parameter permits to control the adaptive generation in every *k* learning step. In each trial, the P_{xyz} vector is the input to the network. The winner centroid ***wij**** is selected by the nearest to end-effector position in terms of Euclidean distance. Then, the value of each weight associated to these centroids will be updated by means of equation (1), starting from random initial values:

$$w_{ji}(k+1) = v_{ij}(k) = \frac{e_i(k) + w_{ji}(k) \times N_j(k)}{N_j(k)+1} \tag{1}$$

where $e_i(k)$ is the ith component of the input vector P_{xyz}; N_j represents the times that jth neuron of F_2 layer has been winner.

The process will be repeated until the convergence of the neuron weights of the ART map is reached. The ρ parameter will be compared, in each iteration, with the Euclidean distance between the new patron position and the winner cell centroid. This comparison will determine the generation of new cells or the updating of computed centroids. In the operation phase, when a target position is detected, the network selects the winner cell. Then, it will project that sensorial over the spatial position of the robot arm, by means of the learnt propioceptive information.

The next step will be to compensate the *DV* between the calculated current position of the robot arm and the desired position in sensorial coordinates. The cell generation permits to know the most favourable posture of the robot arm whose end-effector position is the nearest to the target. By adding the ART algorithm to the neural structure is possible its implementation in any robotic platform with independence of their internal dynamic models.

4 Neural Associative Maps for Sensory-Motor Transformation

The second neural model is dedicated to compensate the error in each cell. Every cell has an independent behaviour for the others, that is, if one cell is excited the others are inhibited. All the cells implement the spatial–rotation transformation. In order to control the robot arm, the neurocontroller must obtain the propioceptive data from the joints and visual information also according to the AVITE learning model from which is inspired. Fig.3. shows the scheme of the learning system, where *TPVs* is the desired spatial position of the arm; *PPVs* is the spatial position of the cell centre; *PPVm* is the angular position of robot arm joints; *DVs* is the difference between *TPVs* and *PPVs;* and *DVm* is the result of the transformation between spatial and rotation increments.

When a cell is excited, the centre of the cell applies its content into PPVm and *PPVs* vector. The *DVs* vector calculates the difference between the centre of the cell and the desired position. The *DVs* is transformed into the *DVm* through a set of neurons. The resulting increments are modulated by a *Go(k)* signal and the results integrated into the *PPVm.*

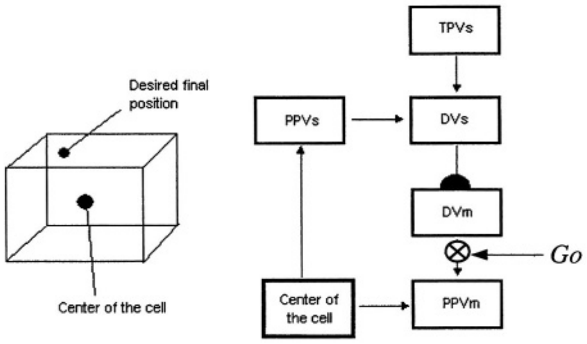


Fig. 3. Structure of the sensory-motor transformation in each learning cell, based on the AVITE fast mapping in spatial (visual) coordinates. The centre of the cell stores the spatial coordinates and the motor coordinates in that point

The learning phase is based in the knowledge acquired in action-reaction cycles. During this phase, random increments are introduced in the *DVm* vector, the system produces these movements and its spatial effect is taken over the *DVs* vector. In this way, the neuron weights, given by ***W*** matrix, are updated by means of the Gradient Descent optimization algorithm. The compensation of the position error produced by the *DV* will be made by the expression (2):

$$\Delta\bar{\theta} = W \cdot \Delta\bar{S} \qquad (2)$$

where $\boldsymbol{\Delta\theta}$ vector computes the incremental values to be added to the current position of the robot arm in spatial coordinates, and $\boldsymbol{\Delta S}$ stores the *DV* in visual coordinates. Each cell generates a neuron weight matrix with a dimension equal to the size of sensorial coordinates (x, y, z) multiplied by the size of spatial coordinates (number of degrees of freedom of the robot arm). Thus, the dimension of ***W*** matrix will be *3xD* for each N^{th} cell, being *N,* the number of the learning cells, and *D* the robot arm d.o.f. The linearity of the equation (4) has the advantage of the easy implementation over a hardware device like *DSP* or *FPGA* and the fast computation of the spatial projection over the motor commands.

5 Results

The implementation of the proposed system has been carried out in a real robotic installation, as Fig.4a shown, formed by an industrial robot arm and the LINCE anthropomorphic stereohead with two colour cameras to simultaneously detect the objective (a small red sphere) and the end-effector robot arm (green label over the gripper). The implementation of the proposed neural architecture has been focused on robotic applications for reaching and tracking targets. The base, elbow and shoulder joints have been considered for the experimentation. Firstly, the generation of cells inside the robot workspace has been carried out based on the described ART neural algorithm. Fig.4b shows a graphical representation of the results for 600 trials and $\rho=0.18$ over the simulation software developed by DISA, Spain [6].

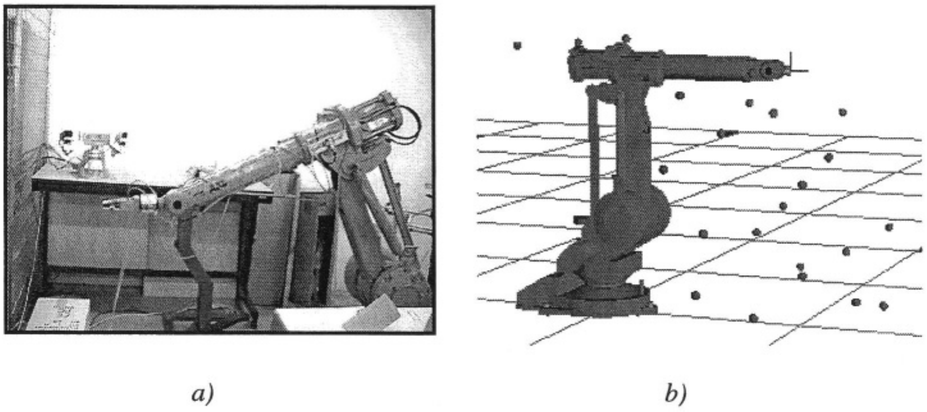


Fig. 4. Robotic installation. (a) Visuo-motor robotic system formed by LINCE stereohead and one ABB-1400 robot arm. (b) Simulation of the centroid distribution by the ART algorithm is shown. For $\rho = 0.18$, 24 centroids have been autonomously generated

To test the proposed model for reaching applications without perturbations several experiments have been carried out. In Table 1, the most relevant results are shown. In it, the behaviour of the model is compared with the number of generated cells and the final error reached, which is given by (3):

$$E(k) = \sqrt{\sum_{i=1}^{3} \left(T\arg et_i(k) - CurrentPosition_i(k)\right)} \quad (mm.) \tag{3}$$

Table 1. Experimental scenarios for reaching tasks. Different ρ parameters, target positions and desired errors have been considered. In all cases Go=1; The end-effector initial position was {900,100,400} and the target positions were T1={100;900;1000} and T2={300;-900,800}. The influence of the ρ parameter in the final precision and in the time-to-reach can be observed

ρ	error (mm)	target (mm)	cells	time (sec.)	ρ	error (mm)	target (mm.)	end-effector final position (mm.)	Time (sec.)
0,12	8,6	T_1	53	3,4	0,13	<1	T_1	{100,5; 900,1;1000,6}	6,0
0,16	7,5	T_1	27	3,4	0,13	<5	T_1	{102,3; 900,5;1002,9}	5,2
0,18	9,4	T_1	21	3,5	0,13	<10	T_1	{104,9; 901,1;1006,2}	4,8
0,12	6,7	T_2	53	3,3	0,13	<1	T_2	{299,6;- 900,0;800,3}	3,2
0,16	7,0	T_2	27	5,5	0,13	<5	T_2	{301,5;- 900,7;799,2}	2,9
0,18	5,9	T_2	21	4,5	0,13	<10	T_2	{294,0;- 900,2;804,1}	2,9

The obtained results for reaching tasks have been compared with other close-loop neural architectures in the same platform [7]. In this case, times to reach the object are reduced about 60%.

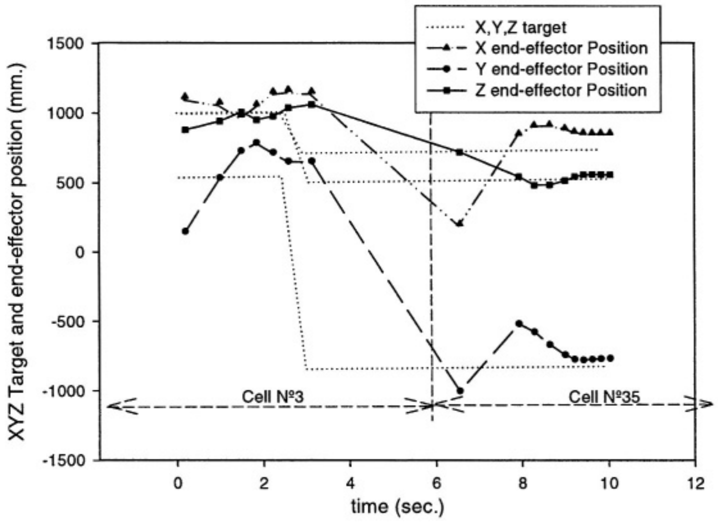


Fig. 6. Evolution of the robot arm end-effector to reach the object with perturbations

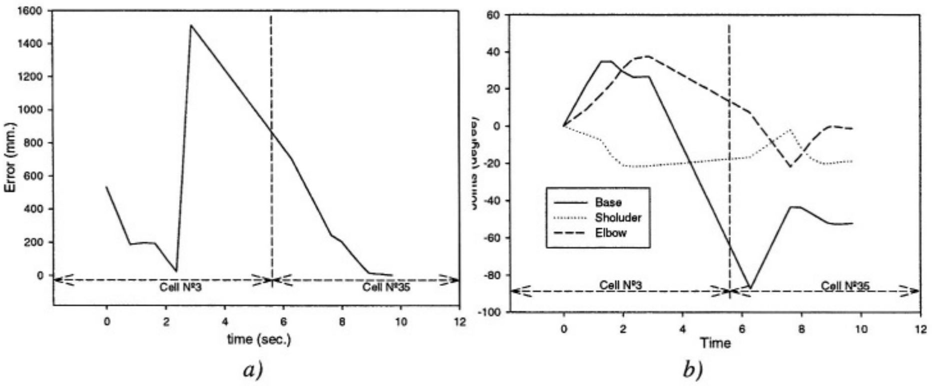


Fig. 7. Evolution of (a) the error and (b) the robot arm joint positions. Because the proposed architecture allows to control the cell commutation when perturbations happen, the error is quickly decreased by means of the open-loop positioning in the centre of every cell, the specific neural weight matrix for each cell and the lineal characteristics of the AVITE model

To test the behaviour of this architecture when unexpected variations in the target position are produced, experiments with instantaneous displacements of the object have been carried out from $P_1=\{1000;500;900\}$ to $P_2=\{700;-900;400\}$. The results are shown in Fig. 6 and 7. In this case, the object position varies from cell N°3 to N°35. Thus, the cell commutation procedure is achieved and the movement compensation inside the second cell is computed by means of the learnt inverse Jacobian matrix which was learnt for that cell.

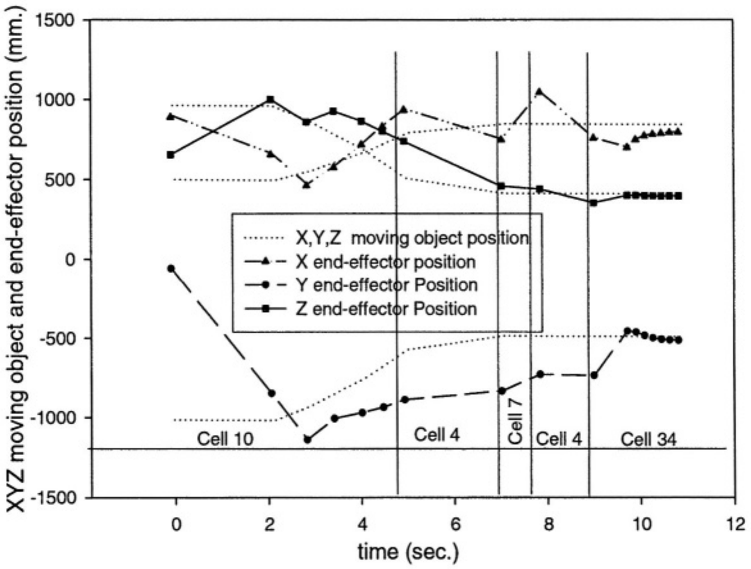


Fig. 8. Evolution of the 3D end-effector position for tracking an object which is moving with constant velocity of 7,6 cm/sec. Five changes of cells are produced and the proposed architecture, quickly compute the next position by means of the associated weight matrix

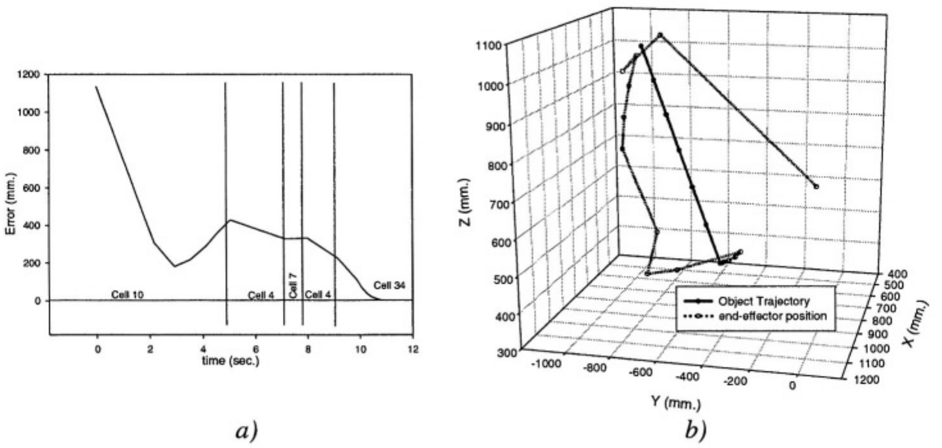


Fig. 9. Evolution of (a) the error in spatial coordinates and (b) the 3D trajectory of the robot arm end-effector and the moving object

Finally, to test the capabilities of the proposed architecture for tracking tasks, constant movements of the object have been generated and the algorithm has been executed. Normally in this case, several commutations of cells are produced and small movement compensations are generated inside each cell of the 3D spatial trajectory. An appropriated filtering in the space of the joints allows to smooth abrupt variations of the end-effector position. Fig. 8 and 9 show the obtained results for tracking tasks.

6 Conclusions

In this paper a neural architecture based on human biological behaviour has been presented and the obtained results have been analysed for robotic reaching and tracking applications with a head-arm system. The 3D spatial division of the robot arm workspace in learning cells is proposed and is solved by means of a self-organizing neural algorithm based on the ART2 model. Indeed, in this process the propioceptive information is learnt. The produced error by the discrepancy between each cell-centre and the target position is compensated by means of an AVITE *(Vector Associative Map)* adaptive architecture. It projects the difference vector of visual position over incremental joint positions of the robot arm. The obtained results over a robotic platform have demonstrated that final error in reaching applications can be very low, taking into account the robustness and fast operation of the model.

Acknowledgments

This work was supported in part by the SENECA Fundation (Spain) PC-MC75/00078/FS/02, and the Spanish Science & Technology Ministry (MCYT) under TIC 2003-08164-C03-03 research project.

References

1. Grossberg, F.H. Guenther, D.Bullock, D.Greve. (1993). “Neural representation for sensory-motor control. II. Learning a head centered visuo-motor representation of 3-D target Positions”, Neural Networks. 6, 43-67.
2. J.Piaget. “The grasp of consciousness: Action and concept in the young child”. Harvard University Press, Cambridge. MA.
3. JL. Pedreño-Molina, A. Guerrero-González, O. Florez-Giraldo, J. Molina-Vilaplana, “Sensory-motor control scheme based on Kohonen Maps and AVITE model”. Artificial Neural Nets. Problem Solving Methods, Part II. J. Springer-Verlag (Lectures Notes on Computer Science), ISSN:3-540-40211-X Mira & J.R. Álvarez (eds), 2003, Vol 2687, pp. 185-192.
4. G.Carpenter, S.Grossberg. “ART 2: Selft-organization of stable category recognition codes for analog input patterns”, Proceedings of the IEEE First International Conference on Neural Networks. Vol.II.pags.727-736.1987
5. D. Bullock, S. Grossberg, “VITE and FLETE: Neural modules for trajectory formation and tension control”. In W. Hershberger, /ed.): Volitional Action, pp.253-297. Amsterdam, North-Holland. 1989.

6. Virtual Robot Simulator. Departamento de Ingeniería de Sistemas y Automática, Universidad Politécnica de Valencia. November 2002.
7. J. Molina-Vilaplana, J.L. Pedreño-Molina, J. López-Coronado, "Hyper RBF model for accurate reaching in redundant robotic systems". Neurocomputing, In press, 2004.

Development of Local Perception-Based Behaviors for a Robotic Soccer Player

Antonio Salim*, Olac Fuentes, and Angélica Muñoz

National Institute of Astrophysics, Optics and Electronics, Luis Enrique Erro # 1, Santa María Tonantzintla, Puebla, 72840, México
{asalimm, fuentes, munoz}@inaoep.mx

Abstract. This paper describes the development of local vision-based behaviors for the robotic soccer domain. The behaviors, which include *finding ball, approaching ball, finding goal, approaching goal, shooting* and *avoiding,* have been designed and implemented using a hierarchical control system. The *avoiding* behavior was learned using the C4.5 rule induction algorithm, the rest of the behaviors were programmed by hand. The object detection system is able to detect the objects of interest at a frame rate of 17 images per second. We compare three pixel classification techniques; one technique is based on color thresholds, another is based on logical AND operations and the last one is based on the artificial life paradigm. Experimental results obtained with a Pioneer 2-DX robot equipped with a single camera, playing on an enclosed soccer field with forward role indicate that the robot operates successfully, scoring goals in 90% of the trials.

1 Introduction

Robotic soccer is a common task for artificial intelligence and robotics research [1]; this task permits the evaluation of various theories, the design of algorithms and agent architectures. This paper focuses on the design and evaluation of perceptual and behavioral control methods for the robotic soccer domain; these methods are based on local perception, because it permits designers to program robust and reliable robotic soccer players that are able to cope with highly dynamic environments such as RoboCup environments.

Vision is the primary sense used by robots in RoboCup. We used a local vision approach with an off-board computer. In this approach, the robot is equipped with a camera and an off-board image processing system determines the commands for the robot. We used this approach because of the advantages that it offers, which include lower power consumption, faster processing and the fact that inexpensive desktop computers can be used instead of specialized vision processing boards. We compare three strategies for pixel classification. One strategy is based on color thresholds [8], another is based on the algorithm of Bruce et al. [6] and the last one is based on the artificial life paradigm.

* The first author is supported by CONACyT, under grant number 171617.

Behaviors were designed and implemented using a hierarchical control system with a memory module for a reactive robotic soccer player [5]. The behaviors, which include *finding ball, approaching ball, finding goal, approaching goal,* and *shooting,* were programmed by hand. The *avoiding* behavior was learned via direct interaction with the environment with the help of a human operator using the C4.5 rule induction algorithm [9].

The paper is organized as follows. Section 2 reviews related work. Section 3 describes the methodological approach used in the design of our robotic soccer player. Section 4 sumarizes the experimental results obtained. Finally, Section 5 discusses conclusions and perspectives.

2 Related Work

2.1 Vision

The cognachrome vision system©, manufactured by Newton Research Labs, is a commercial hardware-based vision system used by several robot soccer teams [13]. Since it is hardware-based, it is faster than software running on a general-purpose processor. Its disadvantages are its high cost and the fact that it only recognizes three different colors.

A number of past RoboCup teams have used alternative color spaces such as HSB or HSV for color discrimination proposed by Asada [2], since these separate color from brightness reducing sensitivity to light variations.

Several RoboCup soccer teams have adopted the use of omnidirectional vision generated by the use of a convex mirror [3]. This type of vision has the advantage of providing a panoramic view of the field, sacrificing image resolution. Moreover, the profiles of the mirrors are designed for a specific task.

The fast and cheap color image segmentation for interactive robots employs region segmentation by color classes [6]. This system has the advantage of being able to classify more than 32 colors using only two logical AND operations and it uses alternative color spaces.

For our vision system, we used the pixel classification technique proposed by Bruce [6] and a variant of the color spaces proposed by Asada [2] (see Section 3.2).

2.2 Control

Takahashi et al. [12] used multi-layered reinforcement learning, decomposing a large state space at the bottom level into several subspaces and merging those subspaces at the higher level. Each module has its own goal state, and it learns to reach the goal maximizing the sum of discounted reward received over time.

Steinbauer et al. [11] used an abstract layer within their control architecture to provide the integration of domain knowledge such as rules, long term planning and strategic decisions. The origin of action planning was a knowledge base that contained explicit domain knowledge used by a planning module to find a sequence of actions that achieves a given goal.

Bonarini et al. [4] developed a behavior management system for fuzzy behavior coordination. Goal-specific strategies are reached by means of conflict resolution among multiple objectives. Behaviors can obtain control over the robot according to fuzzy activation conditions and motivations that reflect the robot's goals and situation.

Gómez et al. [7] used an architecture called dynamic schema hierarchies. In this architecture, the control and the perception are distributed on a schema collection structured in a hierarchy. Perceptual schemas produce information that can be read by motor schemas to generate their outputs.

We used a behavior-based control system or subsumption architecture with a memory module in order to control our robotic soccer player (see Section 3.3).

3 The System

3.1 Hardware and Settings

The robot used in this research is a Pioneer 2-DX mobile robot made by ActivMedia©, equipped with a Pioneer PTZ camera, a manually-adapted fixed gripper and a radio modem. The dimensions of the robot are 44 cm long, 38 cm wide and 34 cm tall, including the video-camera. The robot is remotely controlled by a AMD Athlon 1900 computer with 512 MB of RAM. Figure 1(a) shows a picture of our robotic soccer player.

The environment for the robot is an enclosed playing field with a size of 180 cm in length and 120 cm in width. There was only one goal, painted cyan, centered in one end of the field with a size of 60 cm wide and 50 cm tall. The walls were marked with an auxiliary purple line whose height is 20 cm from the floor. Figure 1(b) shows a picture of the playing field.

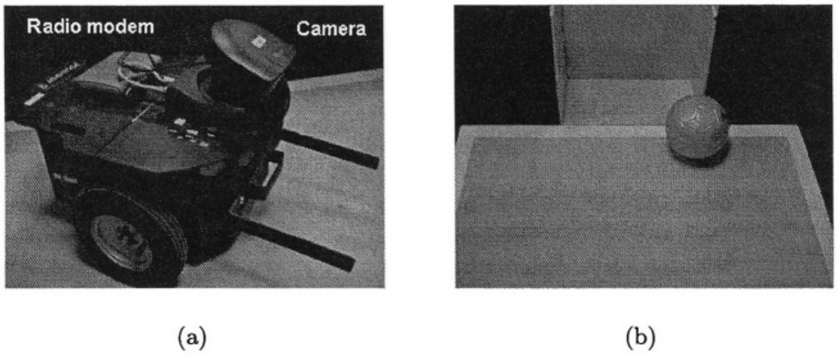


Fig. 1. The robotic soccer player (a). The soccer playing field (b)

3.2 Vision

A robust, fast and fault tolerant vision system is fundamental for the robot, since it is the only source of information about the state of the environment. Since all objects of interest in the environment are colored, we believe that vision is the most appropriate sensor for a robot that has to play soccer. We present below the object detection system used by the robot and a strategy for pixel classification based on the artificial life paradigm.

Object Detection. The vision system processes images captured by the robot's camera and reports the locations of various objects of interest relative to the robot's current location. The objects of interest are the orange ball, the cyan goal and the auxiliary purple line on the field's wall. The steps of our object detection method are:

1. *Image Capture:* Images are captured in RGB in a 160 × 120 resolution.
2. *Image Resizing:* The images are resized to 80 × 60 pixels.
3. *Color Space Transformation:* The RGB images are transformed into the HUV color space, for reducing sensitivity to light variations.
4. *Pixel Classification:* Each pixel is classified by predetermined color thresholds in RGB and HUV color spaces. There are three color classes: the colors of the ball, the goal, and the auxiliary line. The pixel classification is based on [6], in order to use only two logical AND operations for each color space.
5. *Region Segmentation:* Pixels of each color class are grouped together into connected regions.
6. *Object Filtering:* False positives are filtered out via region size.

Figure 2(a) shows an image captured by the frame grabber and Figure 2(b) shows the robot's perception.

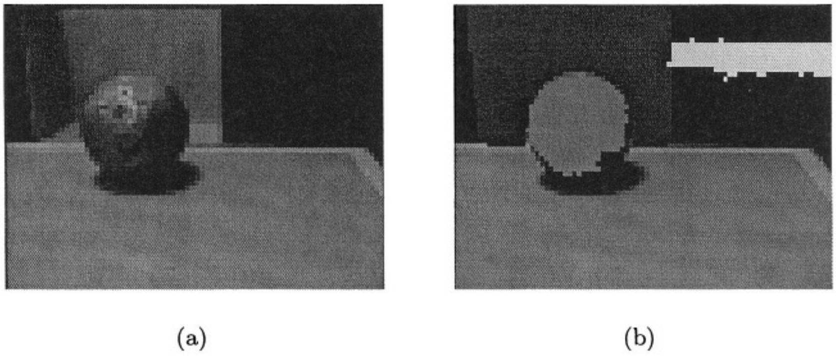


Fig. 2. Image captured by the camera (a). The robot's perception (b)

Artificial Life Approach for Pixel Classification. In order to reduce the time invested in pixel classification, the most expensive step in object detection, we tested an artificial life-based method. Ideas of distributed computing were taken from Reynolds's boids [10], where a group of agents moves as a flock of birds or a school of fish. For this strategy, we used 2500 agents, each having an internal state to indicate whether it is over an object of interest or not. Agents were able to detect three color classes: the colors of the ball, the goal and the auxiliary line in the walls. Agents were serialized by an agent manager which assigned movement turns and prevented collisions between agents. However, the recognition task is distributed among agents. The agents can move in their world, which is the image perceived by the camera. Only one agent can be located over each pixel. Agents can sense the color intensity values in the image in order to perform pixel classification. The locomotion of an agent consists of moving pixel by pixel via its actuators. Figure 3 shows a snapshot of the pixel classification method based on Artificial life.

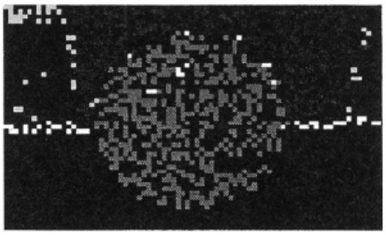


Fig. 3. Artificial life-based pixel classification

3.3 Control

Behaviors were designed and implemented using a subsumption architecture [5] because this architecture offers the necessary reactivity for dynamic environments. We incorporated a new element to this architecture, a memory module. This module acts as a short-term memory that enables the robot to remember past events that can be useful for future decisions. The memory module affects directly the behaviors programmed into the robot.

The *avoiding* behavior is a horizontal behavior in the architecture that overwrites the output of the rest of the behaviors in our vertical subsumption architecture. The architecture was implemented using four threads in C++, one for the vertical behaviors module, one for the memory module, one for controlling the robot movements and one for the horizontal behavior to avoid collisions with the walls. In this architecture, each behavior has its own perceptual input, which is responsible of sensing the objects of interest. Each behavior writes its movement commands to shared memory to be executed. The architecture used for the robot's control system is shown in Figure 4.

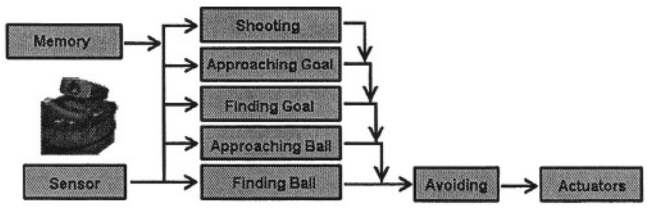


Fig. 4. The architecture of the system

3.4 Description of Modules and Behaviors

1. *Memory:* This is an essential module for the achievement of the robot's global behavior. Memory, like behaviors, has its own perceptual input to sense the ball and the goal. The function of this memory is to remember the last direction in which the ball or the goal were perceived with respect to the point of view of the robot. The memory module affects directly the other behaviors because it writes the directions of the ball and the goal on a shared memory used in the behaviors's execution. There are six possible directions that the memory has to remember: ball to the left, ball to the right, centered ball, goal to the left, goal to the right and centered goal.
2. *Finding Ball:* The robot executes a turn around its rotational axis until the ball is perceived. The robot turns in the direction in which the ball was last perceived. If this information was not registered then the robot executes a random turn towards the left or right.
3. *Approaching Ball:* The robot centers and approaches the ball until the ball is at an approximate distance of 1 cm.
4. *Finding Goal:* The robot executes a turn around its rotational axis until the goal is perceived. The robot turns in the direction in which the goal was last perceived. If this information was not registered then the robot executes a random turn towards the left or right.
5. *Approaching Goal:* The robot executes a turn in the direction of the center of the goal until the goal is centered with respect to the point of view of the robot.
6. *Shooting:* The robot makes an abrupt increase of its velocity to shot the ball towards the goal. There are two possible kind of shots, a short shot when the robot is close to the goal (a distance equal or less than 65 cm) and a long shot, when the robot is far from the goal (more than 65 cm).
7. *Avoiding:* The robot avoids crashing against the walls that surround the soccer field. Determining manually the necessary conditions in which the robot collides with the wall is difficult because the wall can be perceived in many forms, therefore we used the machine learning algorithm C4.5 [9] to learn whether a collision must be avoided or not.

4 Experimental Results

4.1 Pixel Classification Results

We present the results obtained by three implementations of pixel classification. The first implementation was based on color thresholds [8], the second implementation was based on the algorithm proposed by Bruce et al. for pixel classification [6], and finally, the third implementation was based on the artificial life paradigm.

Table 1. Pixel classification results

Method	Images per second	Processing average time
Color thresholds	12 images	0.0874 sec.
Bruce-based method	18 images	0.0553 sec.
Artificial life-based method	14 images	0.0707 sec.

Results of pixel classification are shown in Table 1. As this table indicates, the worst strategy for pixel classification task was based on color thresholds [8]. The best strategy for this task was based on the algorithm proposed by Bruce et al. [6], this strategy was implemented as a step in the object detection system for the robotic soccer player. We expected a better performance from the pixel classification method based on artificial life, because this method needs to examine only 2500 pixels, corresponding to the total number of agents, instead of the total number of pixels in the image (8600 pixels). However, in this strategy each of the agents spends time calculating its next movement, producing a general medium performance.

4.2 Avoiding Behavior Results

For the *avoiding* behavior, we collected a training set of 446 instances of collisions. There were 153 positive samples where there was a collision and 293 negative samples where there was not collision. The elements of the input vector were *roundness, compactness, convexity, orientation, contour length, mean length of runs, line index of lower right corner point, column index of lower right corner point, row of the largest inner circle* and *column of the largest inner circle* of the ball's region detected by the object detection system. The experiments were validated using 10-fold cross-validation. We tested 5 machine learning algorithms for the classification task; the results obtained are summarized in Table 2. As this table shows, the C4.5 algorithm obtained the best percentage of correctly classified instances for the collision avoidance task. The rules generated by C4.5 algorithm were implemented in our *avoiding* behavior.

4.3 Global Performance

Our robotic soccer player has a forward role, thus its main task is to score goals in a minimum amount of time. In order to test the global performance of our

Table 2. Percentage of correctly classified instances by machine learning algorithm for the *avoiding* behavior

Machine learning algorithm	% of correctly classified instances
Support Vector Machines	91.25 ± 0.0603 %
Artificial Neural Networks	90.40 ± 0.0849 %
C4.5	92.20 ± 0.0638 %
Naive Bayes	87.62 ± 0.0683 %
Conjuntive Rules	90.68 ± 0.0222 %

robotic soccer player, we designed a set of experiments. The experiments were performed on the soccer field shown in Figure 1(b). The robot position, robot orientation and ball position were selected 20 times randomly as follows:

1. For selecting the robot's position, the field was divided into 24 cells of equal size. Figure 5(a) shows the cells for the robot position.
2. For selecting the ball's position, the field was divided into 9 cells of equal size. Figure 5(b) shows the cells for the ball position.
3. For selecting the robot's orientation, there were 4 directions to the robot. The orientation where the goal is: 1) in front of the robot, 2) left to the robot, 3) back to the robot and 4) right to the robot. Figure 5(c) shows the possible orientations for the robot.

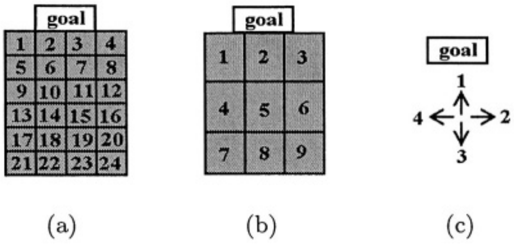


Fig. 5. Experiments's configuration. Robot position (a). Ball position (b). Robot orientation (c)

An experiment's configuration can be represented as a triplet, of the form *(ball position, robot position, robot orientation).* The configuration for the 20 experiments performed were: *(24,7,1), (24,8,1), (21,2,2), (8,8,4), (18,7,3), (22,9,2), (24,4,4), (7,4,3), (6,4,3), (8,2,2), (15,1,3), (21,4,1), (12,2,2), (11,9,1), (7,8,4), (20,9,1), (7,9,4), (11,9,4), (10,5,2)* and *(6,2,3).* Table 3 summarizes the time spent in seconds by each behavior performed by the robot in the experiments. The total time spent by the robot in the experiments was 632 seconds.

The percentage of time used by behaviors in the experiments was 28% for *Finding Ball,* 32.27% for *Approaching Ball,* 14.24% for *Finding Goal,* 9.49% for *Approaching Goal* and 16% for *Shooting.* The average time required by the robot to score a goal is 35.11 seconds.

Table 3. Time spent, in seconds, in each of the behaviors executed by the robot during 20 experiments. The symbol "-" indicates that a behavior in a given experiment was not executed. An experiment that contains only "-" symbols, is an unsuccessful experiment in which the robot got stuck against the walls

Experiment Number	*Finding Ball*	*Approaching Ball*	*Finding Goal*	*Approaching Goal*	*Shooting*	Duration
1	16	10	10	–	6	42
2	11	19	17	1	6	54
3	–	–	–	–	–	–
4	13	9	3	4	5	34
5	8	5	–	2	6	21
6	8	11	7	10	6	42
7	6	31	6	3	6	52
8	5	9	7	1	5	27
9	5	6	–	5	5	21
10	8	6	–	4	5	23
11	2	16	8	–	6	32
12	17	20	11	6	6	60
13	–	–	–	–	–	–
14	–	7	–	3	6	16
15	16	11	–	7	5	39
16	–	5	–	–	6	11
17	27	8	–	4	6	45
18	19	8	–	3	6	36
19	6	12	11	2	5	36
20	10	11	10	5	5	41
Totals	177	204	90	60	101	632 sec

The *avoiding* behavior was successful, the robot avoided 10 of 12 avoidance situations obtaining 83% success.

An useful functionality of the soccer player emerges from the interaction of three behaviors: *approaching ball, finding goal* and *avoiding*. This emergent behavior consist of *regaining the ball from the corner*. In the experiments, the robot was able to regain the ball from the corner four out of five times obtaining 80% success. In the 20 experiments executed, the robot was able to score 18 goals obtaining 90% success.

5 Conclusions

In this paper, we presented our research on the development of local perception-based behaviors for a Pioneer 2-DX robot equipped with a single camera.

The subsumption architecture used for the robot control gives the necessary reactivity to play soccer, also the memory that we incorporated enables the robot to base its decisions on past events.

The *avoidance* behavior was much easier to learn than to program by hand. Building the avoiding behavior using the C4.5 algorithm to learn to avoid collisions with the walls was successful.

Although the strategy for pixel classification based on artificial life did not improve the performance, it seems to be a promising strategy to create a completely distributed control system for a robotic soccer player. The main limitation of this approach is the current computational processing power needed to support a large number of agents with complex behaviors. Using our object detection method we can detect the ball, goal and auxiliary line, at a frame rate of 17 frames per second.

Experimental results obtained with our robotic soccer player indicate that the robot operates successfully showing a high-level intelligent behavior and scoring goals in 90% of the trials.

In future work, we will use other machine learning techniques to help us develop behaviors such as *approaching ball.* The next step to reach in our research is to consider multi-robot coordination, an important issue in robot soccer.

References

1. Asada, M., Stone, P., Kitano, H., Werger, B., Kuniyoshi, Y., Drogoul, A., Duhaut, D., Veloso, M.: The RoboCup Physical Agent Challenge: Phase-I. Applied Artificial Intelligence. 12 (1998) 251–263.
2. Asada, M., Kitano, H.: RoboCup-98: Robot Soccer World Cup II. LNCS, Springer-Verlag. 1604 (1999).
3. Bonarini, A., Aliverti, P., Lucioni, M.: An omnidirectional vision sensor for fast tracking for mobile robots. Trans. on Inst. and Measurement. 49 (2000) 509–512.
4. Bonarini, A., Invernizzi, G., Labella, T. H., Matteucci, M.: An architecture to coordinate fuzzy behaviors to control an autonomous robot. Fuzzy Sets and Systems. 134 (2003) 101–115.
5. Brooks, R. A.: A Robust Layered Control System for a Mobile Robot. IEEE Journal of Robotics and Automation. RA-2 (1986) 14–23.
6. Bruce, J., Balch, T., Veloso, M.: Fast and inexpensive color image segmentation for interactive robots. In Proc. of the 2000 IEEE/RSJ IROS. (2000) 2061–2066.
7. Gómez, V. M., Cañas, J. M., San Martín, F., Mantellán, V.: Vision-based schemas for an autonomous robotic soccer player. In Proc. of IV WAF'2003. (2003) 109–120.
8. Pitas, I.: Digital Image Processing Algorithms and Applications. John Wiley & Sons Inc. (2000).
9. Quinlan, J. R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo, C.A. (1993).
10. Reynolds, C. W.: Flocks, Herds, and Schools: A Distributed Behavioral Model. Computer Graphics. 21 (1987) 25–34.
11. Steinbauer, G., Faschinger, M.: The Mostly Harmless RoboCup Middle Size Team. OGAI. 22 (2003) 8–13.
12. Takahashi, Y., Asada, M.: Vision-guided behavior acquisition of a mobile robot by multi-layered reinforcement learning. IEEE/RSJ IROS. 1 (2000) 395–402.
13. Werger, B. B., Funes, P., Schneider-Fontán, M., Sargent, R., Witty, C., Witty, T.: The Spirit of Bolivia: Complex behavior through minimal control. LNCS, Springer-Verlag. 1395 (1997) 348–356.

Statistical Inference in Mapping and Localization for Mobile Robots

Anita Araneda and Alvaro Soto

Pontificia Universidad Catolica de Chile, Santiago 22, Chile
aaraneda@mat.puc.cl, asoto@ing.puc.cl

Abstract. In this paper we tackle the problem of providing a mobile robot with the ability to build a map of its environment using data gathered during navigation. The data correspond to the locations visited by the robot, obtained through a noisy odometer, and the distances to obstacles from each location, obtained from a noisy laser sensor. The map is represented as an occupancy grid. In this paper, we represent the process using a Graphical Representation based on a statistical structure resembling a Hidden Markov model. We determine the probability distributions involved in this Graphical Representation using a Motion Model, a Perception model, and a set of independent Bernoulli random variables associated with the cells in the occupancy grid forming the map. Our formulation of the problem leads naturally to the estimation of the posterior distribution over the space of possible maps given the data. We exploit a particular factorization of this distribution that allows us to implement an Importance Sampling algorithm. We show the results obtained by this algorithm when applied to a data set obtained by a robot navigating inside an office building type of indoor environment.

1 Introduction

Robust navigation in natural environments is an essential capability of truly autonomous mobile robots. Providing robots with this skill, however, has turn to be a difficult problem. This is particularly true, when robots navigate in unknown environments where globally accurate positioning systems, such as GPS, are not available.

In general, robots need a map of their surrounding and the ability to locate themselves within that map, in order to plan their motion and successfully navigate afterwards. This is why robot mapping and localization is now considered a fundamental component of autonomous mobile robots [18] [10].

Some approaches consider the problem of mapping under the assumption that the locations visited by the robot are known [20]. This situation is not realistic, however, if we consider that sensors of location, such as odometers, carry error in the location measurement. On the other hand, some approaches consider the problem of localization, under the assumption that a map of the environment is available [3] [6] [15]. The actual situation in applications, however,

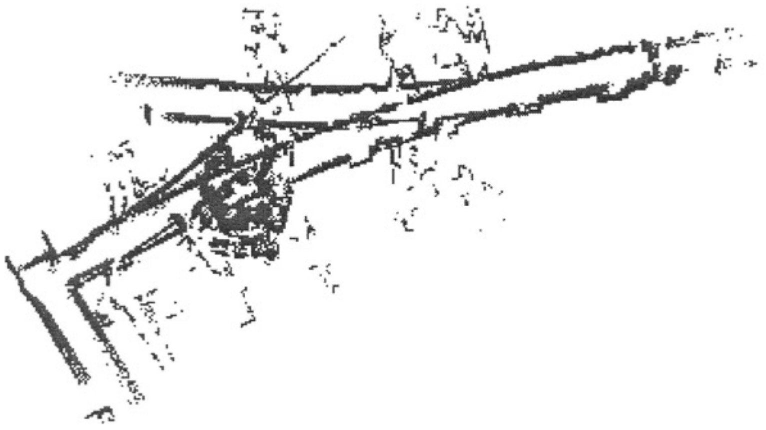


Fig. 1. Map obtained from Raw Data

is that neither the map nor the locations are known. This has led to research on the simultaneous localization and mapping (SLAM) problem using sensor data collected by the robot [13] [21] [19] [10].

As today technology, most of the SLAM approaches consider a robot equipped with an odometer, that collects information about the robot ego-motion, and range sensors, such as sonars or laser range finders, that measure the distances to nearest obstacles. As an example, Figure 1 shows a map drawn from raw odometer and laser readings collected by a mobile robot. The figure shows how odometry error accumulates so that, it seems that the robot has visited two different corridors, instead of just one straight hallway, as it did.

In this paper, we understand SLAM as an estimation problem, where the data correspond to odometer and range sensor readings collected by the robot during its trajectory. The goal is to estimate the posterior distribution of the map and the locations visited by the robot given the data. Odometer readings correspond to rotation and translation measures of the robot movements. Range readings correspond to distances to the closest obstacles, with respect to the robot location. These distances are measured in a set of previously specified directions.

The map is represented by an occupancy grid [5]. In this context, we understand a map of a given environment as a random matrix, each component of which is associated to a spatial location in the environment. The set of associated locations corresponds to a regular grid and each component of the map takes either the value 1 or 0 depending on whether the corresponding location is occupied or not. We express our knowledge of the map through its posterior distribution given the information provided by the robot.

This paper is organized as follows. Section 2 reviews relevant previous work on SLAM. Section 3 discusses the details of our probabilistic approach. Section 4 shows the results of applying our methodology to real data collected by a robot navigating inside an office building type of indoor environment. Finally, Section 5 presents the conclusions of this work.

2 Previous Work

Although there is an extensive research literature touching on mapping or localization for mobile robots[1], the SLAM problem is a relatively newer research area, where most efforts have been made over the last decade. An important family of approaches to SLAM is based on versions of the Kalman filter. The pioneering development in this area is the paper by Smith et al. [16] who basically propose the Hidden Marked Model (HMM) approach widely used today. Smith et al. presents an application of the Kalman filter to the problem of estimating topological maps. They assume a fixed number of landmarks in the environment where these landmarks can be identified by their cartesian coordinates. At a fixed time instant, the set of landmarks coordinates and the location of the robot are assumed to be unobservable or latent variables. As in the Kalman filter, the main assumption is that the posterior distributions of all these variables are Gaussian and that the observations, given the latent variables, can be described as a linear function and a white noise term.

These two assumptions are somewhat restrictive. The Gaussian assumption makes this approach unsuitable for multimodal distributions that arise when the location of the robot is ambiguous. The linearity assumption is not met in general, since the relation between odometry and locations involves trigonometric functions. The Extended Kalman Filter (EKF) [11] partially handles non-linearity using a Taylor approximation.

For the non-Gaussian case, Thrun et al. [22] postulate a general approach that can be used with general distribution functions. Under this approach, however, computing maximum likelihood estimates is computationally too expensive. In [20], Thrun presents an application of the Expectation and Maximization algorithm [4] applied to mapping. The map is treated as the parameter to be estimated while the locations are treated as part of a Hidden Marked Model. Thus, Thrun proposes maximizing the expected log likelihood of the observations and the locations, given the map.

A more recent successful approach to solving the SLAM problem is the FastSLAM algorithm [13]. This approach applies to topological maps, and is based on a factorization of the posterior distribution of maps and locations. The models that determine the process are the ones used in [20].

On the other hand, [9] present an approach that is also based on the description in [20], but this one applied to occupancy grids. This approach finds locations iteratively over time. At each point in time, the algorithm estimates the location visited by the robot as the location that maximizes the probability of the current data, given past data and previous location estimates. Next

[1] See [20] for a good overview of the literature in this area.

step finds the map, as the map that maximizes the posterior probability of the estimated locations and the observed data.

Our mapping approach applies to occupancy grid maps of static environments. Our formulation of the problem is based on the approach by Thrun et al. [22]. We build a Graphical Representation of that formulation where the locations are considered unobservable variables determining the observed odometer readings and, together with the map, determining the observed laser readings. The probability model of the whole process is determined by Motion and Perception Models and a prior distribution for the map.

As opposed to previous approaches, our approach propose a fully Bayesian approach where our goal is to estimate the posterior distribution of the map using simulation. Thus, our approach shows a formal description of the entire process and develops a Bayesian solution. The fact that it uses more general Motion Model than the one used by the Kalman Filter approaches, makes it applicable to a wider set of problems. The advantage of this method is that it does not provide a single estimate of the map, as the EM based solution, but it produces multiple maps showing the notion of variability from the expected posterior map. As for localization, we obtain a simulation of the locations visited by the robot from their posterior distribution, as an intermediate step while simulating maps.

3 Our Approach

We describe the SLAM problem in probabilistic terms. We assume that there are true but unobservable locations visited by the robot and that there are true but unobservable distances to obstacles from each of those locations. These locations and distances determine the map of the environment, represented as an occupancy grid. Odometer and laser readings correspond to the observed values of true locations and distances to obstacles, respectively. We assume that the observations are centered at their true counterparts, having random variations around them[2].

Figure 2a) shows a Graphical Representation of the problem where non-observable variables have been circled for clarity. Our Graphical Representation was developed originally in [1] and there is a similar, but less specific Graphical Representation, developed independently in [14]. The Graphical Representation we develop here has been subsequently adopted in [8].

In Figure 2, U_t represents the difference between odometer readings at times $t-1$ and t. Z_t represents the location of the robot at time t. θ_t represents the distances from Z_t to the closest obstacles in front of the robot at time t. It is important to note that $\boldsymbol{\theta}_t$ is fully determined by the location of the robot, Z_t, and the map, $\boldsymbol{M}$. Finally, V_t represents the laser readings at time t.

Writing $\boldsymbol{U}$ and $\boldsymbol{V}$ to denote the matrices of odometer reading differences and laser readings collected from time 1 to T, the SLAM problem can be expressed

[2] From now on we omit the word "true" when referring to true locations and true distances to obstacles.

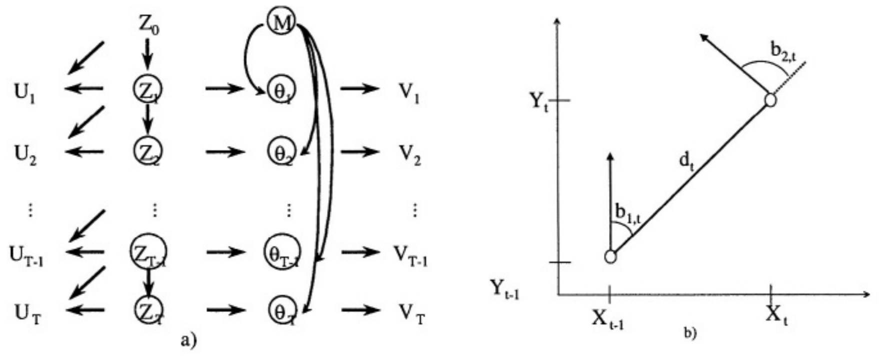


Fig. 2. a) Graphical Representation of the problem. b) Translation and rotation components of movement from Z_{t-1} to Z_t

as the problem of determining the posterior distribution of $\boldsymbol{M}$ given $\boldsymbol{U}$ and $\boldsymbol{V}$, $P(\boldsymbol{M}|\boldsymbol{U},\boldsymbol{V})$.

3.1 Motion and Perception Models

To complete the specification of the process we need to establish the probabilistic models that drive the dependencies, in particular, the so-called Motion and Perception Models. We assume that, to move from Z_{t-1} to Z_t, the robot performs three independent actions: a first rotation , b_{1t}, to face the direction of the translation, a translation , d_t, from Z_{t-1} to Z_t, and a final rotation, b_{2t}, to face the orientation of the robot at time t. These motions are shown in Figure 2b). The elements of U_t correspond to the observed counterparts of $b_{1,t}$, $b_{2,t}$ and d_t, respectively.

Denoting by $\boldsymbol{Z}$ the matrix containing all locations from time 1 to T, and $\boldsymbol{Z}^{t-1}$ the vector of true locations up to time $t-1$, we can write

$$P(\boldsymbol{Z}|\boldsymbol{U}) = \prod_{t=1}^{T} P(Z_t|\boldsymbol{U},\boldsymbol{Z}^{t-1}) = \prod_{t=1}^{T} P(Z_t \mid U_t, Z_{t-1}). \tag{1}$$

The term $P(Z_t \mid U_t, Z_{t-1})$ in the last product in equation (1) is known as the Motion Model. Here we assume a Gaussian Motion Model, which determines that the error of b_{1t}, b_{2t} and d_t with respect to their observed counterparts, correspond to white noise with variances that are proportional to these latter quantities.

Let $\boldsymbol{\theta}$ be the matrix of all distances to obstacles up to time T and $P(\boldsymbol{V} \mid \boldsymbol{\theta})$ be the conditional distribution of laser readings given distances to obstacles. Assuming that laser beams read distances independently from each other and also that these laser readings are independent over time, we write

$$P(\boldsymbol{V}|\boldsymbol{\theta}) = \prod_{t=1}^{T}\prod_{i=1}^{N} P(V_{ti}|\theta_{ti}), \tag{2}$$

where N represents the number of laser beams, and V_{ti} and θ_{ti} represent the laser reading and distance to the closest obstacle, respectively, in the i-th direction at time t. The term $P(V_{ti}|\theta_{ti})$ is known as the Perception Model. We assume that it corresponds to a truncated Gaussian distribution with mean θ_{ti} and with known variance σ^2 determined by the accuracy of the laser sensor. The Gaussian distribution is truncated by 0 at the lower end and the maximum range of the laser at the other.

Finally, we consider that the prior distribution of θ_{ti} corresponds to a geometric distribution, such that $P(\theta_{ti}) = (1-p)^{\theta_{ti}-1}\, p$, where p corresponds to the prior proportion of busy cells in the map. In this work we determine this value empirically by analyzing the data.

3.2 Importance Sampling

In order to find an expression for the posterior distribution of $\boldsymbol{M}$ given $\boldsymbol{U}$ and V we must integrate over pairs $(\boldsymbol{Z}, \boldsymbol{\theta})$, as seen in Figure 2a). Alternatively, we note that the map M is fully determined by $\boldsymbol{Z}$ and $\boldsymbol{\theta}$, thus we are interested in the posterior distribution $P(\boldsymbol{Z}, \boldsymbol{\theta}|\boldsymbol{U}, \boldsymbol{V})$. In the absence of a closed form for this expression, we use Importance Sampling (IS) to sample observations from it.

Importance Sampling [7] [17] is a sampling algorithm used to estimate probability distributions. It has been heavily used in recent years. The main idea in IS is to represent a distribution by a set of observations and a weight associated to each observation in the set. In this way, expected values and other features of the target distribution can be estimated as the weighted average of the observations.

Consider a set of n tuples $(x_j, \omega_j), j = 1, 2, \ldots, n$, given by random draws $x_j's$ from a distribution g and corresponding weights $\omega_j's$. Liu and Chen [12] define this set to be *properly weighted* with respect to the distribution π if

$$\lim_{n\to\infty} \frac{\sum_{j=1}^{n} h(x_j)\omega_j}{\sum_{j=1}^{n} \omega_j} = \mathrm{E}_{\pi}(h(X)),$$

for any integrable function h. This means that if $x_1, x_2, \ldots, x_n$ are a sample from g, the set of weights $\omega_j(x_j) = \pi(x_j)/g(x_j)$ properly weights the sample with respect to π.

3.3 Sampling from the Posterior Distribution of Maps

Our IS approach relies on the factorization of $P(\boldsymbol{Z}, \boldsymbol{\theta}|\boldsymbol{U}, \boldsymbol{V})$ given by $P(\boldsymbol{\theta}|\boldsymbol{U}, \boldsymbol{V})\; P(\boldsymbol{Z}|\boldsymbol{\theta}, \boldsymbol{U}, \boldsymbol{V})$. This decomposition suggests sampling $\boldsymbol{\theta}$ from its posterior distribution, $P(\boldsymbol{\theta}|\boldsymbol{U}, \boldsymbol{V})$, first, and sampling $\boldsymbol{Z}$ from $P(\boldsymbol{Z}|\boldsymbol{\theta}, \boldsymbol{U}, \boldsymbol{V})$ afterwards.

In the case of $P(\boldsymbol{\theta}|\boldsymbol{U}, \boldsymbol{V})$, we approximate this distribution by the product of the Perception Model and the geometric prior distribution of $\boldsymbol{\theta}$. Thus, according to the model described above, we sample values of θ_{ti} from a truncated Gaussian distribution centered at the corresponding laser reading V_{ti} and standard deviation σ, and then we associate to each sample a weight given by $\omega(\theta_{ti}) = (1-p)^{\theta_{ti}-1}\, p$. This set of weights properly weight the set of samples, with respect to the posterior distribution of $\boldsymbol{\theta}$.

Importance sampling plays, again, a key role when sampling locations from $P(\boldsymbol{Z} \mid \boldsymbol{\theta}, \boldsymbol{U}, \boldsymbol{V})$. Using the properties of the model described before, we can show that (see [2] for details)

$$P(\boldsymbol{Z}|\boldsymbol{\theta},\boldsymbol{U},\boldsymbol{V}) \,\dot{\propto}\, \prod_{t=1}^{T} P(Z_t|U_t, Z_{t-1})\; P(\theta_t|Z_t, \boldsymbol{Z}^{t-1}, \boldsymbol{\theta}^{t-1}). \tag{3}$$

Equation (3) shows that, at each point in time, we can sample Z_t from the Motion Model and associate a weight

$$\begin{aligned} \omega(Z_t) &= P(\theta_t|Z_t, \boldsymbol{Z}^{t-1}, \boldsymbol{\theta}^{t-1}) \\ &= \prod_{i=1}^{N} P(\theta_{ti}|Z_t, \boldsymbol{Z}^{t-1}, \boldsymbol{\theta}^{t-1}), \end{aligned} \tag{4}$$

to this observation. The term $P(\theta_t|Z_t, \boldsymbol{Z}^{t-1}, \boldsymbol{\theta}^{t-1})$ corresponds to a truncated geometric distribution that represents the degree of agreement between true distances to obstacles at time t, θ_t, and the fact that the robot is at the sampled location, Z_t, within the map built from $\boldsymbol{Z}^{t-1}$ and $\boldsymbol{\theta}^{t-1}$.

4 Results

In this section we show an application of the algorithm described in previous section, to a data set obtained in Wean Hall building at Carnegie Mellon University. The data set was collected by a robot equipped with a laser sensor and an odometer[3].

The robot navigated going back and forth along a hallway. In that journey 3354 measurements were taken, each of them consisting in a pair of odometer reading differences and laser readings. The laser sensor sends beams every degree spanning an angle of 180°. Thus, there are N = 180 distances recorded for each laser reading. A map drawn from raw data is shown in Figure 1. The figure shows how odometry error accumulates so that it seems that the robot has visited two different corridors, instead of just one, as it did. The smoothness of the depicted walls, however, suggests that error in laser sensor readings is small, compared to error in odometer readings.

We sampled distances to obstacles based on the Perception Model, with $\hat{\sigma} = 0.02m$, and sampled locations afterwards, using IS with sample size $n = 100$. Figure 3 shows a path obtained from a set of sampled locations, along with

[3] Data is a courtesy of Nicholas Roy.

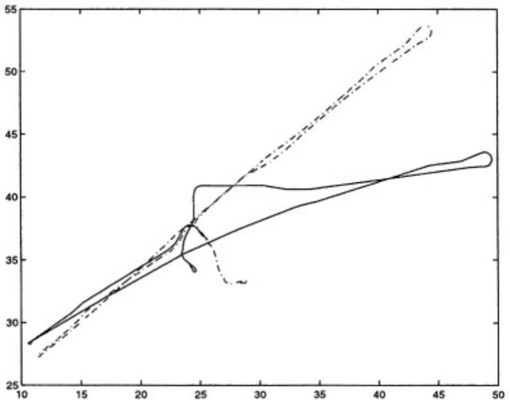


Fig. 3. Path of the robot in Raw data (line) and Sampled Path (dotted)

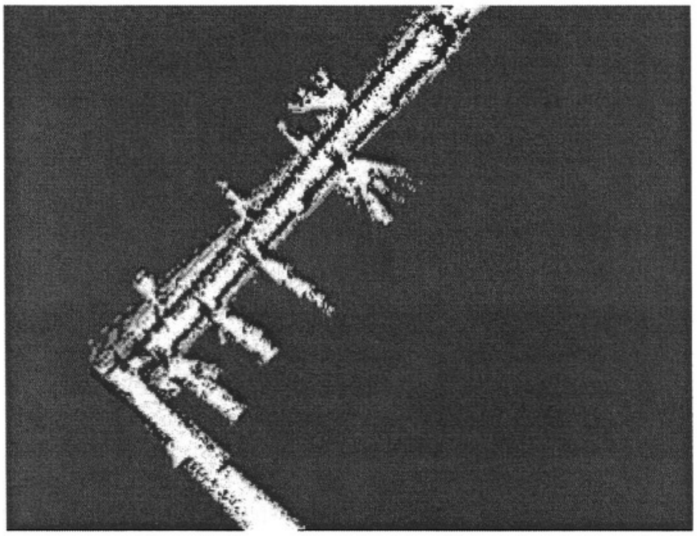


Fig. 4. Average map using the proposed Algorithm

the path obtained from raw odometer readings. The figure shows that sampling using IS allows the robot to recover from odometry errors. Figure 4 shows the average map of the sample.

5 Conclusions

This paper adds to the research in Statistics and Probabilistic Robotics, proposing a complete probabilistic representation of the SLAM problem and obtaining a full Bayesian solution. In particular, it contributes with a new algorithm to sample from the posterior distribution of maps. An intermediate step of the algorithm provides observations from the posterior distribution of locations, that is, we solve the localization problem at the same time.

This paper formalizes the problem of mapping as the problem of learning the posterior distribution of the map given the data. We work on an expression for this distribution and show that there is no closed form for it. Thus, we propose an algorithm based on Importance Sampling for obtaining a sample from the target posterior distribution.

Important Sampling showed to be a computational efficient way to explore the posterior distribution of the map. In addition to this, IS provided with an effective methodology to correct odometry error accumulated over time.

Although we do not have ground truth data to quantify the accuracy of the algorithm, the average of the resulting map samples closely resembles the real map of the environment. In the same way, samples from the robot trajectory resemble the true path followed by the robot.

References

1. A. Araneda. *Statistical Inference in Mapping and Localization for Mobile Robots.* PhD thesis, Thesis Proposal. Dept. of Statistics, Carnegie Mellon University, 2000.
2. A. Araneda. *Statistical Inference in Mapping and Localization for Mobile Robots.* PhD thesis, Department of Statistics, Carnegie Mellon University, 2004.
3. W. Burgard, A.B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. Experiences with and interactive museum tour-guide robot. *Artificial Intelligence,* 114:3–55, 1999.
4. A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society, Series B,* 39:1–38, 1977.
5. A. Elfes. *Occupancy Grids: A Probabilistic Framework for Robot Perception and Navigation.* PhD thesis, Department of Electrical and Computer Engineering, Carnegie Mellon University, 1989.
6. D. Fox, W. Burgard, and S. Thrun. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research,* 11, 1999.
7. J. Geweke. Bayesian inference in econometric models using Monte Carlo integration. *Econometrica,* 57:1317–1339, 1989.
8. D. Hähnel, D. Fox, W. Burgard, and S. Thron. An efficient FLOTSAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. 2002.
9. D. Hahnel, D. Schulz, and Burgard. W. Map building with mobile robots in populated environments. 2002.
10. P. Jensfelt, H.I. Christensen, and G. Zunino. Integrated systems for Mapping and Localization. IEEE, May 2002.
11. S.J. Julier and J.K. Uhlmann. A new extension of the Kalman Filter to nonlinear systems. 1997.
12. J. Liu and R. Chen. Sequential Monte Carlo methods for dynamic systems. *Journal of the American Statistical Association,* 93, 1998.

13. M. Montemerlo and S. Thrun. Simultaneous localization and mapping with unknown data association using FastSLAM. Submitted for publication, 2002.
14. K. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning.* PhD thesis, Computer Science Division. U. C. Berkeley, 2002.
15. R. Simmons and S. Koenig. Probabilistic robot navigation in partially observable environments. pages 1080–1087, 1995.
16. Randall Smith, Matthew Self, and Peter Cheeseman. *Autonomous Robot Vehicles,* chapter Estimating Uncertain Spatial Relationships in Robotics, pages 167–193. Springer-Verlag, 1990.
17. M. Tanner. *Tools for Statistical Inference.* New York: Springer-Verlag, 1996.
18. S. Thrun. Exploration and model building in mobile robot domains. pages 175–180, 1993.
19. S. Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence,* 99:21–71, 1998.
20. S. Thrun. Robotic mapping: A survey. Morgan Kaufmann, 2002.
21. S. Thrun, M. Beetz, M. Bennewitz, W. Burgard, A.B. Cremers, F. Dellaert, D. Fox, D. Hähnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz. Probabilistic algorithms and the interactive museum tour-guide robot Minerva. *International Journal of Robotics Research,* 19(11):972–999, 2000.
22. Sebastian Thrun, Wolfram Burgard, and Dieter Fox. A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning and Autonomous Robots,* 31/5:1–25, 1998. Joint Issue.

Fusing a Laser Range Finder and a Stereo Vision System to Detect Obstacles in 3D

Leonardo Romero, Adrián Núñez, Sergio Bravo, and Luis E. Gamboa

Universidad Michoacana de San Nicolás de Hidalgo , Morelia, Mich., 52000, Mexico
lromero@zeus.umich.mx,m_anunez@faraday.fie.umich.mx
{legg, sbravo}lsc.fie.umich.mx

Abstract. A new method to detect 3D Obstacles using a stereo vision system and a 2D laser range finder is presented. Laser range finder measures distance to obstacles, but only on a plane parallel to the floor; and stereo vision is not able to estimate distances to surfaces with little or no texture at all. This paper explores a form to take advantages of both kind of sensors. The main idea is to project 3D points detected by the laser telemeter in the form of initial values for pixel disparities into a trinocular stereo vision system. Experimental tests using a mobile robot in a indoor environment showed promising results.

1 Introduction

Most robotic tasks require to detect obstacles around the robot. Recent techniques in obstacle detection and mapping using single-plain laser range finders on mobile robots have been successfully applied in indoor environments [6]. Stereo vision systems also have been applied to 2D [5] and 3D mapping [1]. The disadvantage of a 2-D laser range finder is that only detects obstacles in a plane parallel to the floor and its behavior depends upon the reflectivity of objetcs. Stereo vision also have disadvantages. A successful stereo system must solve the correspondence and the reconstruction problem [7]. The correspondence problem is to find which parts of the images taken from the cameras are projections of the same scene element. The reconstruction problem involves to compute the 3-D location and structure of the observed objects. Sometimes the correspondence problem can not be solved or it is very difficult problem. For example, large homogeneous regions are hard elements to find its correspondence. To overcome this problem, some approaches process only vertical lines or edges [5].

However, it is often useful in mobile robotics to fuse data from multiple, possible heterogeneous sensors. A fusion of sensor data can happen at the data, the feature, or the decision level [3]. Data fusion occurs when the raw data from various sensors are combined without significant post-processing. Feature fusion occurs when features are extracted from data at the sensor level before combination. Fusion at the decision level requires that the fusser algorithm accumulate data at the sensor level before fusing. An algorithm to fuse, at the feature and data level, range data from stereo vision and lidar sensor is described in [3]. They use an occupancy grid map (the environment is divided in regular regions and each region has an occupancy status) to fuse data from sensors. Other work fuse data from stereo vision and sonars [5] mixing maps obtained separately by each sensor.

The goal of the present work is to combine, in a different way, a 2-D laser range finder into a trinocular stereo vision system to form a more robust 3-D obstacle detection system. The key idea is to exploit the range information captured by the 2-D laser range in order to solve some ambiguities of the stereo vision system and get a better disparity map [7], and also to speed up stereo calculations.

This paper introduces a variant of the Shirai's Algorithm (described in [2]), a correlation method using variable window, where laser measurements are included in the algorithm in the form of initial disparity values. In this way some of the problems of stereo vision are solved (e.g. Homogeneous surfaces). To build a disparity map, images taken from cameras are rectified using the calibration method described in [4]. Using rectified images, the search for correspondences is reduced to one dimension.

The rest of the paper is organized as follows. Section 2 shows the image rectification process. Sections 3, 4 describes the mapping from laser measurements to image coordinates. Sections 5 and 6 shows the combination of the laser data into the stereo processing. Some experimental results using a mobile robot are shown in section 7.

2 Stereo Rectification

In a standard stereo system where the cameras have parallel optical axis and their image planes are orthogonal to the optical axis, the correspondence search process is reduced to a search within the same pixel row. The process of converting images taken with a given geometry to those corresponding to a standard stereo system is called rectification [7].

In [4] a method based in an image register technique is used to accomplish the distortion correction. The method matches two images: a distorted image acquired by the camera and a calibrated pattern without distortion. This process applied to several cameras can (having certain mechanical considerations) built a standard stereo system, given that the cameras end up capturing the same calibration pattern.

The rectification process we use is the following:

1. Mechanically approximate the stereo geometry to a standard geometry. In Figure 1(a) the stereo geometry of the cameras is shown. A strategy that worked for us is putting a pattern of three dots distributed in a way that corresponds to the stereo geometry (see Figure 1(b)), and then mechanically align the geometrical center of the cameras with their counter parting reference dot.
2. Put the calibration pattern over a plane Π that is orthogonal to the projection axis of the cameras in an initial position such that it is aligned and centered with the left camera and acquire the image.

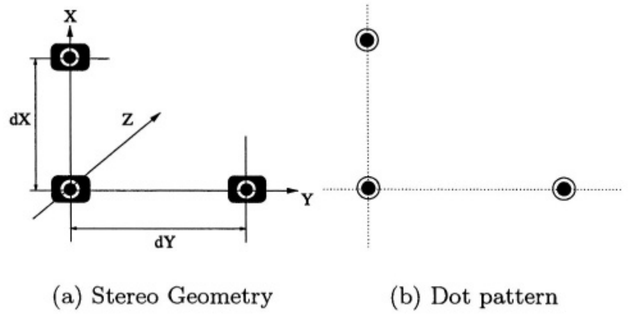


Fig. 1. Process of image rectification

3. Slide *dX* the calibration pattern over the *Π* plane and acquire a second image with the topmost camera.
4. Slide *dY* the calibration pattern from its initial position and acquire a third image with the right camera.
5. Finally, apply the calibration method of [4] using the same calibration pattern (see Fig. 5(a)) and each acquired image.

The process generates three correction files, one for each camera, that allows us to convert the images to their corresponding corrected and rectified images (see Figure 2 for an example of the left and right cameras).

3 Integrating the Laser Telemeter with the Stereo Geometry

Using a reference frame centered on the left camera, as shown in Fig. 1(a), the laser plane is mechanically aligned as shown in Fig. 3. It is straightforward to map points measured by the laser sensor to the reference frame.

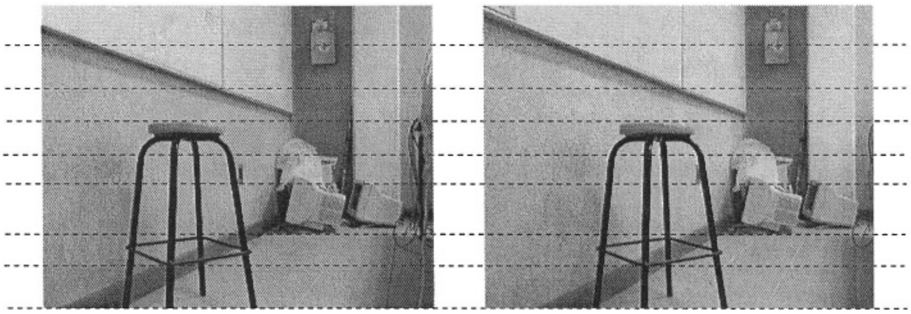


Fig. 2. Rectified left and right images

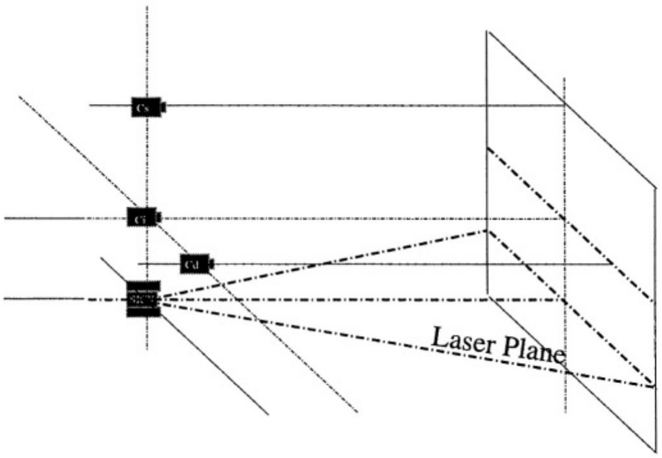


Fig. 3. Integrating the laser with the stereo geometry

4 Mapping Laser Reads to Images

Given that the lens distortion has been corrected and the images are rectified we can use the perspective model (*pinhole* model) [7], which allows us to convert 3D coordinates to image coordinates. The perspective model assumes no distortion, so the coordinates (x_i, y_i) of a laser point or measurement with coordinates (X_i, Y_i, Z_i), within the left image are given by:

$$x_i = f\frac{X_i}{Z_i}, \quad y_i = f\frac{Y_i}{Z_i}$$

where f is the focal distance.

5 Laser Data Propagation

Information obtained from the laser is going to be used as disparity seeds. Disparity d_i is obtained by [7] $d_i = \frac{f\,dY}{Z_i}$, (see Fig. 1 (a) for *dY*). Our main goal is for these seeds to transmit their disparity to their neighbors and so on. Yet, each pixel in the neighborhood of a pixel with a disparity value should adopt such a disparity, only if the pixel pass the following tests,

- *Texture Test* - Pixel texture should be similar to the texture of their neighbors.
- *Horizontal Correspondence Test* - We try to find a match between the given pixel (in the left image) and its respective pixel in the right image, by using the left and the right images, as well as the disparity information d_i stored in the neighboring pixel.
- *Vertical Correspondence Test* - It is similar to the previous step, but now we use the information from the top and bottom images.

Each pixel inherits the disparity pixel of its neighbor just if it has passed the previous tests. To expand the disparity seeds through the whole image we apply a breadth first search. At the beginning only pixels with disparity values computed by the laser data are in the list of nodes to explore. Each node taken from the list is expanded to its eight neighboring pixels.

6 Correlation Method

In order to complement our propagation method, we use Shirai's approach [7], a correlation method that assesses disparity between detected borders within stereo images. This algorithm is based on the fact that detected borders are more suitable to match than other homogeneous areas. Furthermore, this criterion decreases computational cost. Shirai's algorithm uses dynamical-length windows. Hence, narrow and wide windows are used in large and shorter searching regions, respectively. Image gradient is used as a border detector, which is given by,

$$|\nabla I(x,y)| = \sqrt{(\frac{\partial I(x,y)}{\partial x})^2 + (\frac{\partial I(x,y)}{\partial y})^2} \tag{1}$$

A border has been detected if $\nabla I(x,y) > T_\nabla$, where T_∇ is a previously defined threshold.

The following proposed criteria, within the Shirai's algorithm, aim to find the right correspondence.

- All the matches detected with Shirai's algorithm are stored.
- The mean μ error and standard deviation σ are obtained from the previous stored matches.
- If a minimum match is located far enough from the other ones, say 2σ at least, it is considered to be a suitable correspondence.

A similar process is carried out to find suitable correspondences with the third image.

Disparity maps are built in a excluding way by using the proposed propagation described in the previous section and the modified Shirai's method. Hence, the propagation method is used within homogeneous regions, and Shirai's method is activated within detected borders regions.

7 Experimental Results

Experiments were performed by using three a mobile robot with three Firewire cameras (resolution 640 × 480 with 2.1mm lens) and a SICK Laser Measurement System model LMS209-S02. The laser sensor covers a 180° range with a 0.5° resolution and detects a maximum distance of $32m$. Since our method requires

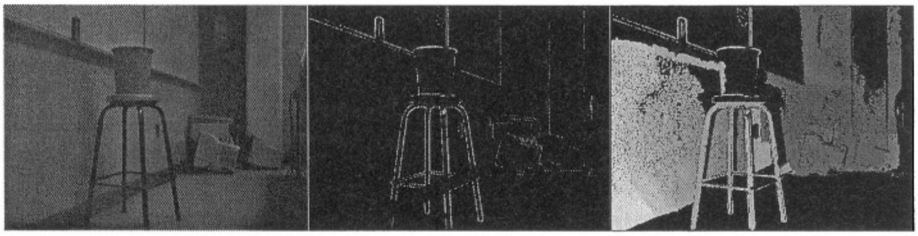


Fig. 4. Results obtained from stereo images and laser data

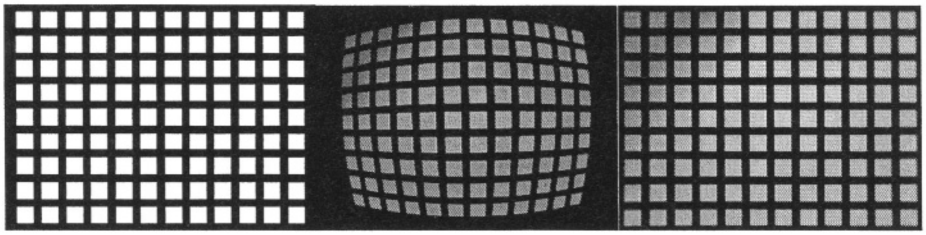


Fig. 5. Correcting image distortion by using a register technique

non-distorted images, we first apply an extension[1] of the algorithm shown in [4]. Figure 5 draws an example of correcting the image distortion produced by the cameras. The synthetic calibration pattern is shown in Fig. 5(a), the image captured by the camera is shown in Fig. 5(b), and the image obtained from calibration process is shown in 5(c), which shows a very good result.

Figure 4 shows an application of the proposed method. Figure 4(b) shows the Shirai's method response when a maximum window length of 7 pixels, and a searching region of 80 pixels is used. In this case, initial parameter $T_{\nabla} = 16$ and 2σ were used.

The disparity map obtained from mixing both of the previous methods, correlation and propagations is shown in Fig. 4(c). In this case, the similarity threshold among neighbors was 16 gray tones. It is possible to observe that this process better detects homogeneous regions.

[1] The original approach uses k_1 and k_2 parameters to model distortion, which was not enough to correct distortion of lens with 2.1 mm focal distance. We include another parameter k_3.

8 Conclusions

We have presented a method that combines a laser range finder and a stereo vision system to build better disparity maps. Fusing only laser data and stereo range data does not include information from homogeneous surfaces above or below the laser plane.

The proposed approach takes advantage of including the laser data into the trinocular stereo vision system and works fine with homogeneous surfaces. More tests are going to be done as well as developing a probabilistic propagation instead of the deterministic propagation described. The idea is to include conditional probabilities of disparities given by the laser sensor, the stereo vision system and disparities of neighboring pixels.

References

1. L. Iocchi, K. Konolige, and M. Bajracharya. Visually realistic mapping of a planar environment with stereo. In *Proc. of Seventh International Symposium on Experimental Robotics (ISER),* Hawaii, 2000.
2. Reinhard Klette, Karsten Schlns, and Andreas Koschan. *Computer Vision, Three-dimensional Data from Images.* Springer-Singapore, 1998.
3. Kevin M. Nickels, Andres Cata o, and Christopher Cianci. Fusion of lidar and stereo range for mobile robots. In *Proceedings of the International Conference on Advanced Robotics,* 2003.
4. Leonardo Romero and Felix Calderon. Correcting radial lens distortion using image and point correspondences. In *Proceedings of the 8th Iberoamerican Congress on Pattern Recognition (CIARP),* La Habana, Cuba, 2003. Springer.
5. S. Thrun, A. Bucken, W. Burgard, et al. Map learning and high-speed navigation in rhino. In D. Kortenkamp, R. P. Bonasso, and R Murphy, editors, *Artificial Intelligence and Mobile Robots,* pages 21–52. AAAI Press/The MIT Press, 1998.
6. S. Thrun, W. Burgard, and D. Fox. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3d mapping. In *Proceedings of the International Conference on Robotics and Automation (ICRA),* 2000.
7. E. Trucco and A. Verri. *Introductory Techniques for 3-D Computer Vision.* Prentice Hall, 1998.

Adaptive Automata for Mapping Unknown Environments by Mobile Robots

Miguel Angelo Abreu de Sousa, André Riyuiti Hirakawa, and João José Neto

Escola Politécnica da Universidade de São Paulo,
Departamento de Engenharia de Computação e Sistemas Digitais,
Av. Prof. Luciano Gualberto, travessa 3, no.158, São Paulo - SP - Brasil - CEP 05508-900
{miguel.sousa, andre.hirakawa, joao.jose}@poli.usp.br

Abstract. Robotic mapping is one of the most important requirements for a truly autonomous mobile robot. Mobile robots should be able to building abstract representation of the physical environment, in order to navigate and work in such environment. This paper presents an adaptive way to make such representation. The proposed system allows the robot to explore all the environment and acquire the information incoming from the sensors (presence or absence of obstacles) while it travels. The robot may start the mapping process at any point of the space to be mapped. Due to the adaptability of the chosen method, the process has the capability of dynamically increase the memory requirements according to the already mapped area, even without any a priori knowledge of the environment.

Keywords: Adaptive Automata, Robotics, Navigation and Robotic Mapping.

1 Introduction

Early approaches for allowing mobile robots to move around used to employ a preliminary map of the environment stored in its memory. Those approaches do not provide an adequate solution since storing a complete geometrical map of the environment, searching the database for localization and the path planning process significantly increases the computational complexity of the system, making the approaches prohibitive for actual implementations [2].

Another problem related to those approaches refers to the repetitive nature of non-automatic mapping processes. Each unstructured environment in which the robot is intend to work, such as, buildings, offices, industries and agricultural fields, has to be mapped first and the resulting map must be manually registered in its memory.

There is also a question related to the possibility of robots to work in hazardous and unknown environments. Dangerous tasks like mining, undersea operations, working in disaster areas, space and planetary exploration are examples of situations in which robots may have to map the field before being able to work properly.

Stimulated by such reasons, robotic mapping has been a strongly researched topic in robotics and artificial intelligence for two decades, and still presenting challenging research subjects, e. g. mapping dynamic or large areas [9].

The present work proposes an adaptive mechanism to steer a robot to cover all its unknown environment and, during this exploration, to collect information from its sensors and to organize them as a map. This map is built in such a way that robot's navigation become easier.

2 Related Work

Since the early 80's robotic mapping research area has been split between metric and topological approaches. Metric maps represent the environment by using its geometric properties [4] [8]. Topological maps describe environments as a set of important places, which are connected by arcs [2] [6]. These arcs have attached information on how to navigate through such places. Nevertheless, the exact frontier between these approaches has always been fuzzy, since topological maps rely on geometric information about the world [9].

Adaptive devices change their structure and behavior according to their external stimulus. Such feature represents an intuitive and trustful way for modeling physical environments and to conduct the robot, despite the complexity of the environment.

2.1 Adaptive Automata

Adaptive automata, first proposed in [7], extends the concept of finite automata by incorporating the feature of performing dynamic self-reconfiguration in response to externally collected information. Such behavior provides adaptive automata with learning capability, which makes them suitable for representing knowledge.

It has been shown that adaptive automata are Turing-powerful devices [7] and they have also been applied on several applications, such as pattern recognition [3] and systems description [1].

Adaptive automata may be viewed as self-modifying state machines whose structure includes a set of states and a set of transitions interconnecting such states. States may be classified in: initial state; a set of final states; and a set of intermediate states. Incoming stimuli change the internal state of the machine.

The self-modifying feature of adaptive automata is due to the capability it has of changing its own set of transition rules. Adaptive actions may be attached to the transitions which are able to either add new states and transitions or remove already existents ones, consequently, achieving a new structure. Hence, incoming stimuli may change the set of internal states and modify the general configuration of the automaton. See [7] for details on concepts and notation.

Transition rules in adaptive automata are represented as:

$$(g, e, a) : B \rightarrow (g', e', a') : A$$

g: push-down store contents before the transition;
g': push-down store contents after the transition;

e: current state before the transition;
e’: current state after the transition;
a: input stimulus before the transition;
a’: input stimulus after the transition;
B: adaptive action before applying the transition;
A: adaptive action after applying the transition.

Adaptive actions A and B are both optional. Three different elementary adaptive actions are allowed: inspection – search the current state set for a given transition; deletion – erase a given transition from the current state set; and insertion – add a given transition to the current set of states. Such actions are denoted by preceding the desired transition by the signs ?, – and +, respectively. Figure 1 shows a graphic representation of the transition.

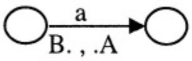


Fig. 1. Simplified transition **(e , a) : B → e’ : A** when g, g’ and a’ are omitted

2.2 The Mapping Automaton

The initial work on representing physical environments by using adaptive automata has been proposed in [5]. The present paper extends this work.

The adaptive mapping automaton starts from a square lattice (figure 2a) consisting of nine states connected by special transitions, all of them denoting areas to be mapped. The central state is the initial state of the automaton, and represents the starting point of the exploring path.

In order to allow a clear presentation of the method, a graphic representation of the adaptive automaton will be used (The dot-marked state corresponds to the actual position of the robot and single lines represent areas not yet mapped.).

Initial automaton also presents special tags (X), marking corner states, and special transitions, which are provided for supporting expansions in the lattice, as shown in figure 2b.

The automaton properly replaces the four adjacent non-filled transitions according to the data information collected by the robot’s sensors while it performs the exploring moves. The information collected by the sensors contains indications on the direction – north, south, east or west – and the condition of the place – free or busy.

Figure 2c shows one possible example of the four-data information collected by sensors and stored by the automaton. Double arrows indicate non-obstructed areas and bold lines denote obstructed ways.

In order to exemplify a complete map, figure 3a illustrates the representation of the information acquired by the automaton after exploring a simple room. The dot-marked state shows that the robot completed the exploration at the rightmost upper space of the room.

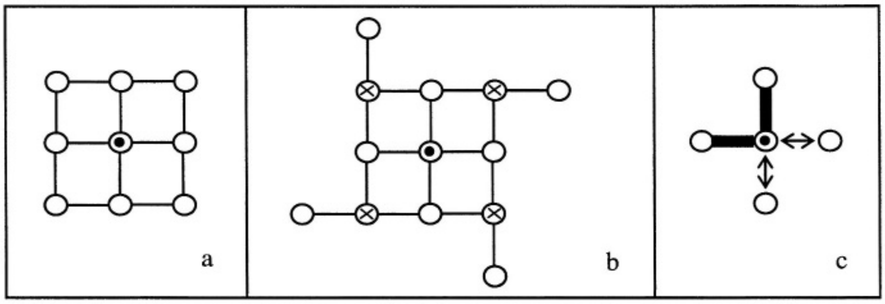


Fig. 2. (a) Initial lattice in an adaptive automaton. (b) Special tags and transitions in initial automaton. (c) Example of information coming from the sensors: two directions obstructed and two free directions

To keep the relation between the part already built of the map and the real environment, the initial state of the automaton is adopted as representing the origin of the map. This state corresponds to the initial mapping place. So, any point in the map is associated to any point in the physical environment through the association of each transition in the map's representation to some corresponding displacement performed by the robot in the real world.

Note that in actual applications the automaton is represented in the algebraic way (as described in section 2.1).

This mapping process allows dynamic memory occupation usage according to the amount of already mapped area. This feature contrasts with classic approaches, such as those described in [4], [6], [10].

3 The Model

The proposed model to perform robotic mapping and exploring motion by using adaptive automata is depicted in figure 4.

In this proposal, an information management system supplies the exploring-motion automaton with data collected from the sensors, and the current neighborhood information previously modeled in the mapping automaton. Data collected by the sensors contain information on the direction (north, south, east or west) and condition (occupied or free). Data from the map contain information on the two adjacent states in the four directions and their condition (free, occupied, not mapped or not created state). Figure 3b illustrates an example of information from the map representing the two south states in an occupied condition and all other states in a free condition.

The output information generated by this automaton indicates in which direction the robot is going to move. According to the configuration of the environment, the exploring-motion automaton also presents in its output a landmark information for future assistance to the navigation.

The management system supplies the mapping automaton with information collected from sensors followed by the direction information and the landmark – in case of occurrence. The mapping subsystem is responsible to store all the sensor information on the presence or absence of obstacles close to the robot and the navigation auxiliary landmarks. The motion decision is also transferred to the motion system that controls the robot's motors.

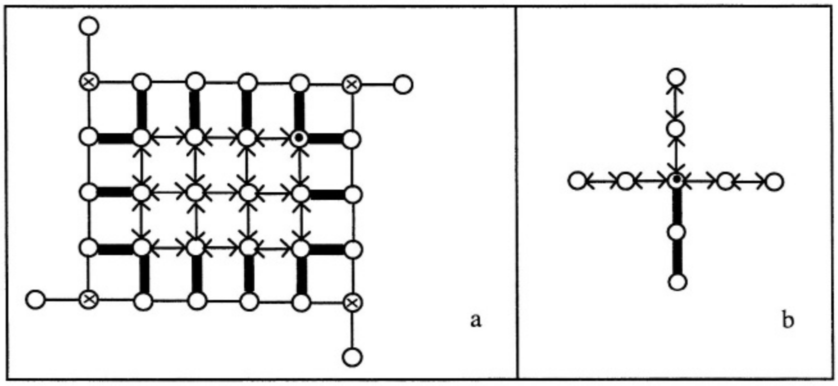


Fig. 3. (a) Example of a simple room mapped. (b) Information extracted from the map

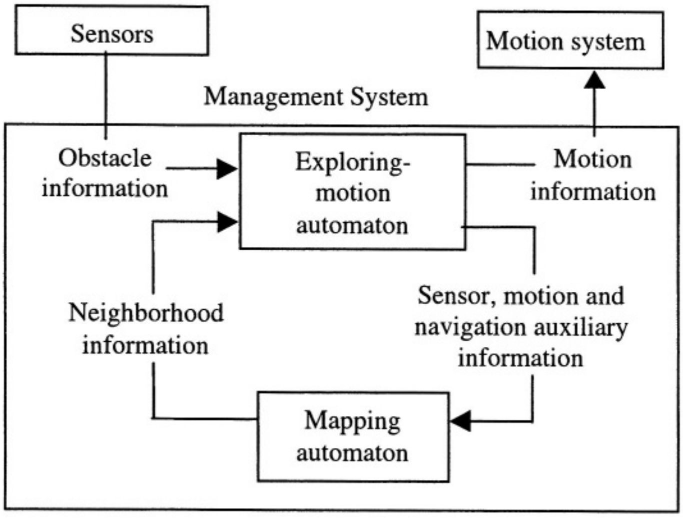


Fig. 4. System model

Note that the environment, sensors and motions have been simulated in order to validate the proposed map building mechanism.

4 The Exploring-Motion Automaton

As described in section 4, an adaptive automaton is used for determining the robot's next move. For this purpose, it is supplied with information collected by the sensors and with neighborhood information previously registered in the map. Its operation allows the robot to cover the whole environment by describing a zigzag path. This section illustrates some details of this procedure through the following example:

a. Starting at any point of the environment, the automaton leads the robot to north direction until it finds an obstacle (Figure 5a).
b. The exploring-motion automaton conducts the robot from north to south until it finds an obstacle. Then, it turns around and comes back in a parallel path, describing a zig-zag, which grows up to east (Figure 5b).
c. After each step towards east and before the robot comes back in the parallel path, the automaton searches for a free space "behind" the robot, filling this space (in case of occurrence), and then returning to its zig-zag path (Figure 5c).
d. When this east-growing zig-zag path is exhausted, the robot returns backwards by the same trajectory searching for a sequence of "free – not mapped" adjacent states at east or west (for this purpose the automaton uses information extracted from the map as shown in figure 3b). Such sequence means that there is a non-mapped space in that this direction and the way is free to reach it. Note that such space may be a simple place or another environment as complex as the already mapped one (Figure 5d).
e. When such new space is located at the eastern side of the robot, it explores this space by performing the zig-zag path, as described in step 'b' (Figure 5e).
f. When such new space is located at the western side of the robot, it explores this space by performing the zig-zag path, but grows to the west only after assuring that the sequence of "free – not mapped" adjacent states still exist at this direction. A sequence of "free – free" adjacent states means that this space has already been mapped and the automaton leads the robot backwards, as described in step 'd' (Figure 5f).
g. When the robot returns to its initial point of exploration, the automaton leads it to the same sequence described above, changing 'east' to 'west', i. e., the zig-zag path grows to west until exhausted, and displacements to the east are conditioned to occurrence of a sequence of "free – non-mapped" adjacent states at this direction (Figure 5g).
h. When the robot returns again to its initial point of exploration, the automaton signs that the environment is entirely explored and full (Figure 5h).

The zig-zag path has been chosen by its generalist features and because the environment is completely unknown. Some approaches divide the environment in rectangles during the exploration. In fact, a zig-zag path may be interpreted as a rectangle with unitary side and whose inside is completely known, which is an important feature in the mapping process.

When a new space is found during the exploration it is entirely explored before proceeding. Such a depth search has been chosen because breadth search implies increasing the actual motion of the robot, then, exploring an entire branch before moving to another one is usually cheaper.

While the environment is explored, the exploring-motion automaton may sign to the mapping automaton some special states, or landmarks, which are properly marked on the map. During the navigation process such landmarks are helpful for plan a trajectory from some initial position to a target position. The system calculates the path between such landmarks and, during navigation, it must find which landmarks are nearest to the initial and to the target positions [12], [13]. Then, those landmarks may be viewed as sub-goals in the navigation process. In order to implement such sub-goals, the exploring-motion automaton searches obstacles to the east or to the west during the zig-zag. If an obstacle is detected (a wall for instance) the proposed automaton marks the central state on the free space before and/or after the wall and, during the return trail, it signs to the mapping automaton that this state is a sub-goal (Figure 5i).

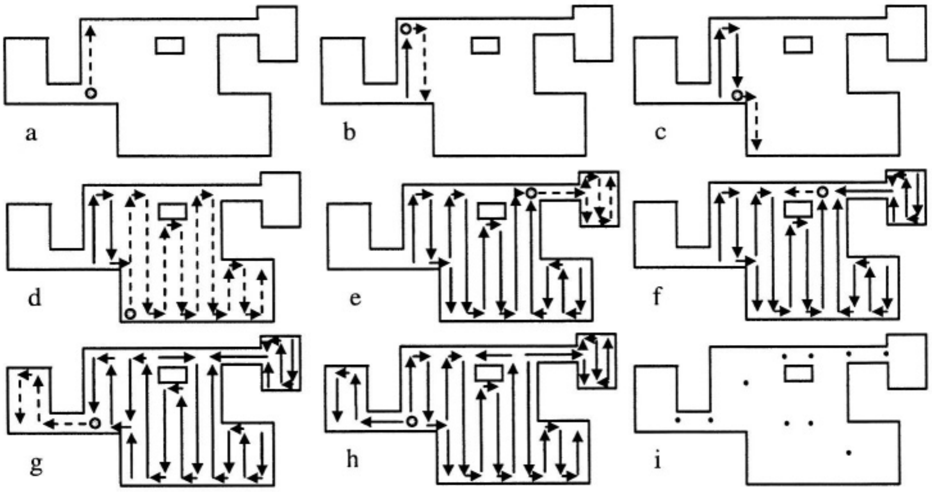


Fig. 5. (a) Initial north move. (b) East-growing zig-zag path. (c) Space filling before complete zig-zag path. (d) Exhausted east-growing zig-zag path. (e) East new space found during the return move. (f) West new space found during the return move. (g) Exhausted west-growing zig-zag path. (h) Environment entire explored. (i) Landmarks defined for the environment-example

The present proposal allows the robot to cover all environment despite to its complexity and to start at any point of the space to be mapped. These features are advantages if compared to other approaches (such as presented in [11]).

4.1 Description of the Automaton

For short, the description of the motion of the exploring automaton is represented in table 2, which expresses the relation between the system's situations of exploration (directions) and the incoming sensor information and the information encoded in the map. Table 1 shows the encoding of table 2.

Table 1. Encoding of table 2

Action	Move to:	Direction	Action	Move to:	Direction
A	North	1.	J	north	6
B	south	2	K	west	7
C	east	3	L	south	8
E	final of exploration	-	M	west	9
F	movement of return	10	N	south	3
G	movement of return	11	O	north	5
H	north	4	Q	north	7
I	east	5	R	south	9

Direction 1: Initial north	N: north
Direction 2: South east-growing	S: south
Direction 3: South complementary east-growing	E: east
Direction 4: North east-growing	W: west
Direction 5: North complementary east-growing	
Direction 6: North west-growing	Ac: action
Direction 7: North complementary west-growing	
Direction 8: South west-growing	↑: free
Direction 9: South complementary west-growing	∅: occupied
Direction 10: Return from east	
Direction 11: Return from west	

Table 2. Situations of exploration versus the incoming information from sensors and map

Directions	Enlargements allowed to																			
	east and to west					east					west					none				
	N	S	E	W	Ac	N	S	E	W	Ac	N	S	E	W	Ac	N	S	E	W	Ac
1	↑				A	↑				A	↑				A	↑				A
	∅	↑			B	∅	↑			B	∅	↑			B	∅	↑			B
	∅	∅	↑		C	∅	∅	↑		C	∅	∅	↑		C	∅	∅	↑		C
	∅	∅	∅	↑	K	∅	∅	∅	↑	K	∅	∅	∅	↑	K	∅	∅	∅	↑	K
	∅	∅	∅	∅	E	∅	∅	∅	∅	E	∅	∅	∅	∅	E	∅	∅	∅	∅	E
2		↑			B		↑			B		↑			B		↑			B
		∅	↑		C		∅	↑		C		∅		↑	M		∅			F
		∅	∅	↑	M		∅	∅		F		∅		∅	F					
		∅	∅	∅	F															

Table 2 (*continuation*)

Directions	Enlargements allowed to																			
	east and to west					east					west					none				
	N	S	E	W	Ac	N	S	E	W	Ac	N	S	E	W	Ac	N	S	E	W	Ac
3		↑			N		↑			N		↑			N		↑			N
	↑	∅			H	↑	∅			H	↑	∅			H	↑	∅			H
	∅	∅	↑		C	∅	∅	↑		C	∅	∅			F	∅	∅			F
	∅	∅	∅		F	∅	∅	∅		F										
4	↑				H	↑				H	↑				H	↑				H
	∅		↑		I	∅		↑		I	∅			↑	K	∅				F
	∅		∅	↑	K	∅		∅		F	∅			∅	F					
	∅		∅	∅	F															
5	↑				O	↑				O	↑				O	↑				O
	∅	↑			B	∅	↑			B	∅	↑			B	∅	↑			B
	∅	∅	↑		I	∅	∅	↑		I	∅	∅			F	∅	∅			F
	∅	∅	∅		F	∅	∅	∅		F										
6	↑				J	↑				J	↑				J	↑				J
	∅			↑	K	∅		↑		I	∅			↑	K	∅				G
	∅		↑	∅	I	∅		∅		G	∅			∅	G					
	∅		∅	∅	G															
7	↑				Q	↑				Q	↑				Q	↑				O
	∅	↑			L	∅	↑			L	∅	↑			L	∅	↑			B
	∅	∅		↑	K	∅	∅			F	∅	∅		↑	K	∅	∅			G
	∅	∅		∅	G						∅	∅		∅	G					
8		↑			L		↑			L		↑			L		↑			L
		∅		↑	M		∅	↑		C		∅		↑	M		∅			G
		∅	↑	∅	C		∅	∅		G		∅		∅	G					
		∅	∅	∅	G															
9		↑			R		↑			R		↑			R		↑			R
	↑	∅			J	↑	∅			J	↑	∅			J	↑	∅			J
	∅	∅		↑	M	∅	∅			G	∅	∅		↑	M	∅	∅			G
	∅	∅		∅	G						∅	∅		∅	G					
10			↑		C			↑		C				↑	K					F
			∅	↑	K			∅		F				∅	F					
			∅	∅	F															
11				↑	C			↑		C				↑	K					G
			↑	∅	K			∅		G				∅	G					
			∅	∅	G															

5 Conclusion and Future Work

Robotic mapping is an essential feature to allow robots to complete certain tasks in unstructured and unknown environments. This work has shown an alternative to the classic mapping approaches: adaptive algorithms provide a new way to build maps and conduct the robot for unknown environments, covering all space. During the exploration, sensors attached to the robot scan the environment for the presence or absence of close obstacles, and such information is collected into the model by enabling the automaton to perform appropriate self-modifications. The exploring-motion automaton also provides special landmarks to the map, which may be used as subgoals on further navigation. The present proposal has the advantage of allowing the robot to explore complex environments without a priori knowledge of the place and the advantage of memory space usage increasing with the actually mapped area.

Future works should approach the navigation problem by using the landmarks and the map created.

References

1. Almeida Jr., J. R.; Neto, J. J. "Using Adaptive Models for Systems Description, Applied Modelling and Simulation", International Conference on Applied Modelling and Simulation, Cairns, Australia, 1999.
2. Brooks, R. A. "Visual Map Making for a Mobile Robot", Proceedings IEEE Int. Conference Robotics and Automation., St. Louis, MO, 1985, pp. 824–829.
3. Costa, E. R.; Hirakawa, A. R.; Neto, J. J. "An Adaptive Alternative for Syntactic Pattern Recognition", Proceeding of 3rd Int. Symposium on Robotics and Automation, ISRA 2002, Toluca, Mexico, 2002, pp. 409-413.
4. Donald, B. R.; Howell J. "Practical Mobile Robot Self-Localization", Proceedings International Conference on Robotics and Automation, San Francisco, CA, 2000.
5. Hirakawa, A. R; Júnior, J. R. A.; Neto, J. J. "Adaptive Automata for Independent Autonomous Navigation in Unknown Environment", International Conference on Applied Simulation and Modelling, Banff, Alberta, 2000.
6. Jennings, J.; Kirkwood-Watts, C.; Tanis, C. "Distributed Map-Making and Navigation in Dynamic Environments", In Proceedings of the International Conference on Intelligent Robots and Systems IROS'98, Victoria, Canada, 1998.
7. Neto, J. J. "Adaptive Automata for Context-Dependent Languages", ACM SIGPLAN Notices, 1994, v.29, n.9, pp.115-24.
8. Sukhatme, G. S.; Wolf, D. F. "Towards Mapping Dynamic Environments", Proceedings of the International Conference on Advanced Robotics, 2003, pp. 594-600.
9. Thrun, S. "Robotic Mapping: A Survey", Exploring Artificial Intelligence in the New Millenium, Morgan Kaufmann, 2002.
10. Zimmer, U. R. "Embedding local map patches in a globally consistent topological map", Proceedings of the International Symposium on Underwater Technology, Tokyo, Japan, 2000, pp.301-305.
11. Yang, S.; Luo, C.; Meng, M. "Entire Region Filling in Indoor Environments using Neural Networks", Proceedings of the 4° World Congress on Intelligent Control and Automation, Shangai, China, 2002, pp.2039-2044.
12. Ma, C.; Liu, L.; Li, W. "Mobile Robot Motion by Integration of Low-Level Behavior Control and High-Level Global Planning", Proc. of the IEEE Int. Conference on System, Man and Cybernetics, China, 1996, pp.310-315.
13. Innocenti, C. et al. "Trajectory planning and real-time control of an autonomous mobile robot equipped with vision and ultrasonic sensors", Proc. of the Int. Conference on Intelligent Robots and Systems IROS'94, Munich, September 12-16, 1994, pp.1861-1866.

Digital Image Processing of Functional Magnetic Resonance Images to Identify Stereo-Sensitive Cortical Regions Using Dynamic Global Stimuli

Héctor-Gabriel Acosta-Mesa[1], Nicandro Cruz-Ramírez[1], John Frisby[2], Ying Zheng[2], David Buckley[3], Janet Morris[3], and John Mayhew[2]

[1] Faculty of Physics and Artificial Intelligence, Department of Artificial Intelligence, University of Veracruz, Sebastián Camacho # 5, 91000, Xalapa, Ver. México
{heacosta, ncruz}@uv.mx

[2] Artificial Intelligence Vision Research Unit, Department of Psychology, University of Sheffield, Western Bank, S10 2TP, Sheffield, UK
{j.p.frisby, ying.zheng, j.e.mayhew}@Sheffield.ac.uk

[3] Section of Academic Radiology, Royal Hallamshire Hospital, University of Sheffield, Glossop Road, S10 2JF, Sheffield, UK
{d.buckle, janet.morris}@sheffield.ac.uk

Abstract. Functional magnetic resonance images (fMRI) were analyzed to investigate the cortical regions involved in stereoscopic vision using red/green anaglyphs to present random dot stereograms. Two experiments were conducted both of which required high attentional demands. In the first experiment the subjects were instructed to follow the path of a square defined by depth and moving in the horizontal plane contrasted with a similar sized square defined by a slight difference in luminance. Three main regions were identified V3A, V3B and BA7. To test that the observed activations were not produced by the pursuit eye movements, a second experiment required the subjects to fixate whilst a shape was presented in different random orientations. Our results suggests that areas V1, V3A and precuneus are involved in stereo disparity processing. We hypothesise that the activation of the V3B region was produced by the second order motion component induced by the spatio-temporal changes in disparity.

1 Introduction

Although many psychophysical studies have investigated how the human brain computes stereoscopic information, it is uncertain which cortical areas are involved in its implementation. Some electrophysiological studies in monkeys report the sensitivity of V1 to absolute disparities, suggesting that this area could be a preliminary stage of processing for stereo information [1]. MT/V5 in monkeys shows a columnar organisation tuned for disparity [2]. MT/V5 in human brains has been widely reported as a motion sensitive area [3-5]. Given the similarity between the visual system of the monkey and the human, it is possible that V5 in human brains is involved in the processing of stereo information.

Recently, modern non-invasive neuroimaging techniques, eg PET and fMRI, have been used to explore the functional anatomy of stereoscopic vision in humans. The results of these studies suggest that stereo disparity processing maybe widespread over a network of cortical regions in the occipital and parietal lobes, including V1, V2, V3, V3A, V3B, [6-10]. However there is no general agreement about the cortical regions selective to stereo disparities or the specific role that each of these has in the perception of depth [11]. The main goal of the present study was to process functional magnetic resonance images to investigate the cortical regions sensitive to stereo disparities using a stereo stimulus that avoids adaptation and at the same time maximises the attentional demands. Under this principle, two experiments were developed using functional magnetic resonance imaging to identify stereo sensitive regions stimulated by random dot anaglyph stereograms.

2 Materials and Methods

Ten healthy subjects, nine right-handed and one left-handed volunteers (7 female, 3 male) aged from 20 to 30 years participated in the first experiment and five healthy right-handed subjects (2 female, 3 male) aged from 20 to 30 years participated in the second experiment. All subjects gave informed written consent. The stereo acuity of the subjects was measured using a stereo vision test (RANDOT SO-002), all of them were below 40 sec of arc. The subjects were given a preliminary practice session outside the magnet to become familiar with the visual stimulation.

2.1 Stimulus Presentation

Subjects lay on their backs in the magnet. They wore red/green anaglyph glasses and looked via a mirror angled at ~45o from their visual axes at a back illuminated screen located just outside the magnet. The viewing distance was 2.4 m. Stimuli were projected on to the screen using an EPSON (EMP-7300) projector driven by a 3G Mac running Psychophysics Tool Box ver. 2.44 [12, 13] under MATLAB ver. 5.3. Although the stimuli were displayed at a video frame rate of 60 Hz, the image was only updated on every 10th frame, producing an effective frame rate of 6 Hz.

2.2 Data Acquisition

Subjects were scanned in a 1.5 T whole-body MRI scanner (Eclips Marconi Systems) with BOLD contrast echo planar imaging (TR= 3s, TE= 40 ms, 128 x128 voxel, voxel size 1.875 x 1.875 x 4 mm.). Thirty two slices covering the whole brain were acquired.

2.3 Data Analysis

The data was pre-processed and analysed using SPM99 (Welcome Department of Cognitive Neurology). The first five scans of each run were discarded to exclude magnet saturation artefacts. All volumes were slice timed, motion corrected and normalised in the MNI (Montreal Neurological Institute) stereotaxic space. The data were smoothed using a 6 mm FWHM (full width at half maximum) isotropic Gaussian kernel. Data analysis was performed using a boxcar design matrix of the different conditions convolved with the hemodynamic response function. Specific effects were tested by applying the corresponding linear contrast to the parameters obtained applying General Linear Model (GLM) using the design matrix [14]. The statistical parametric maps (SPMs) were then interpreted in the standard way with reference to the probabilistic behaviour of Gaussian random fields [17]. The threshold adopted was $P < 0.05$ (corrected). Due to technical restrictions, it was not possible to develop retinotopic mapping to identify the visual areas activated in our studies, instead, the regions of activation identified through the statistical analysis were mapped to anatomical locations using as a reference their Talairach coordinates. The labels assigned to each region were given, matching the anatomical location reported by other authors.

3 Experiment 1: Global Stereo Tracking

Experiment 1 was designed to activate stereo sensitive regions by requiring the subjects to perform a task of global stereo tracking (GST). Random dot stereograms were used to define a square region moving horizontally in front of the background from left to right and vice versa. The subjects were instructed to perform pursuit eye movement to follow the path of the square with their eyes. The performance of the task depended on the maintenance of the perception of depth defined by stereo disparities. The paper which explains in detail this experiment was submitted for publications in the Mexican International Conference in Computer Science 2004 (ENC04).

3.1 Experiment Design

Subjects were given a sequence of 4 scans (sequences) each lasting 5.15 min. (10 epoch) with a 5 min. interscan interval to permit subjects to rest. One hundred image volumes were obtained in each sequence. Each condition lasted 30s. giving 10 multislice volumes per condition (TR=3s.). Each scan consisted in alternating epochs in a boxcar configuration. A dummy condition of a blank screen was presented during the first 15s. (5 scans) of each run which were excluded to control for magnetic saturation effects. The display contained 1,024 dots (with radius 0.1 deg. and zero disparity) distributed over the screen (mean dot density 1.5 dot deg-2). The subjects were instructed to fixate on the right superior corner of a square (5.23 deg. side long) moving laterally across the screen (13 deg. field of view).

The square was moved from left to right and vice versa at a constant speed (2.19 deg.sec-1), each time that the square reached one edge of the screen it changed its direction. Dynamic random noise was used in order to remove any motion cues introduced by the change in disparity [8]. Two modalities were used to define the square, each one representing an experimental condition: a) Two dimensional tracking (2D): The square was luminance defined, its luminance (8.56 cd/m2) was lower than the background (18 cd/m2). This condition was used as a base line. b) Three dimensional tracking (3D): The square was depth defined (red/green anaglyph stereogram), presented 0.3 deg at the front of the background (zero disparity). The square moved horizontally in the plane X/Y, not in plane Z (motion in depth). Stereo-sensitive regions were identified by comparing the activation during the 2D control condition with the neural activity during the 3D stereo tracking condition. Although we could not track the eye movements to monitor the subject's performance of the task, we have no reason to believe that there was any difference between them in the two experimental conditions. After both the training session and the scan session all the subjects reported they could easily track the path of the square in both the 2D and 3D conditions.

3.2 Results: Global Stereo Tracking

Stereo-sensitive regions were identified using a contrast that compared the activation produced by the moving square defined by depth with that produced by the moving square defined by luminance. The results show consistent activation in areas V3A, V3B and parietal cortex. Area V3A in the right hemisphere was activated by depth in 7 of the 10 subjects. In two of these subjects there was bilateral activation. Area V3B in the right hemisphere was activated in 8 of the 10 subjects. In five of these there was bilateral activation. One subject showed activation only in left V3B. Area BA7 (precuneus) in the right hemisphere was activated in 3 of the 10 subjects. Two subjects had left hemisphere activation. There was no bilateral activation. Activation in the right superior parietal area was found in 2 other subjects. It is important to clarify that reversing the contrast revealed no across subject consistent regions in which the activations of the 2D tracking condition were greater than the 3D stereo tracking (data not shown). Results of previous fMRI studies of eye movements have suggested the involvement of parietal areas in the control of pursuit and saccadic eye movements [17-18]. None report the involvement of V3A and V3B regions. In this study, despite the control in which the eye-tracking component was identical in the two conditions, we find an increased activation in the parietal cortex in the stereo tracking task in several of the subjects. This suggests some involvement of the parietal region in stereo-processing per se independent of eye movements. This will be examined further in Experiment 2.

4 Experiment 2: Shape Discrimination from Stereopsis

In order to avoid possible artefacts introduced by eye movements, in Experiment 2 the stereo sensitive regions were explored using a task which performance depended on the continuous perception of stereo depth without requiring tracking. A pie graph (pacman) shape defined either by luminance (2D) or by depth (3D) was displayed at the centre of the screen. The figure changed to one of four possible positions every second. The subjects were instructed to fixate a central dot and press a button when they identified a certain position of the figure. This task provided a way to assess how well the subjects are performing the task.

4.1 Experiment Design

The stimulation sequences used in this experiment were similar to those used in experiment 1. The display contained 4,761 dots (radius 0.04 deg.) distributed over the screen (mean dot density 7 dot deg-2). A pacman shape (radius 4.3 deg.) was displayed at the centre of the screen. The pacman shape was presented in one of four possible positions (up, down, left, right) randomly changing every second. The change in position was constrained to prevent a shape being shown continuously for more than one second in any one position. The positions of the dots were changed between frames (in both conditions) to produce the effect of dynamic random noise [8] to remove any possibility of there being monocular shape cues in the stereo condition:

The subjects were instructed to fixate a point (0.3 deg. of radius and zero disparity) in the middle of the screen (circular field of view 13 deg.) and press a button with the right hand when the mouth of the pacman was in the up position. The response on the button box was recorded to assess the performance of the subjects. There were two modalities to define the pacman, each one represents one experimental condition. a) Luminance (2D): The pacman was luminance defined, its luminance (9.29 cd/m2) was lower than the background (17.7 cd/m2). Both the background and the pacman were placed in the same plane (zero disparity).

This condition was used as a base line. b) Depth (3D): The pacman was depth defined (red/green anaglyph stereogram), and appeared in front (-0.076 deg. of disparity) of the background (0.076 deg. of disparity). The pacman and the background were displayed in front and behind the fixation point respectively in order to remove possible shape cues introduced by the red/green stereoscopic pair of dots in the stereo condition. As in the previous experiment, the stereo-sensitive regions were identified using a contrast which subtracted the neural activity during the control condition from the neural activity during stereo condition. The performance of the task was assessed integrating all the occasions in which the subjects pressed the button at the right time (when the pacman was in up position). Then, the occurrences when the subjects pressed the button at wrong times (when the pacman was not in up position) were subtracted.

4.2 Results: Shape Discrimination from Stereopsis

The stereo-sensitive regions were identified by comparing the activation produced by the pacman defined by depth with that produced by the pacman defined by luminance. Consistent with our expectations V3A, V3B and precuneus were activated. Activations were also found in striate cortex (V1). Area V3A was activated bilaterally in 3 of the 5 subjects, and in the right hemisphere of 2 subjects. Consistent with experiment 1, area V3B was activated bilaterally in 1 subject, and in the right hemisphere of 4 subjects. The precuneus was activated bilaterally in 1 subject and in the right hemisphere of 4 subjects. The V1 region was activated bilaterally in 2 subjects, in the left hemisphere of 2 subjects and in the right hemisphere of 1 subject. Figure 1 shows results from a representative subject, and Tables 1, 2, 3 and 4 shows the MNI coordinates of regions of activation for all the subjects.

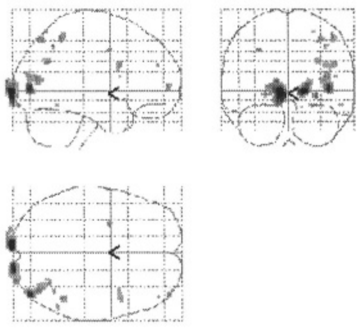


Fig. 1. Depth against Luminance. The statistical map shows the areas sensitive to stereoscopic information. The activation includes V1, V3A, V3B and precuneus

Table 1. Stereo sensitive region: V3A

Subject	*Location*	*Z-Score*	*P corrected*	*Cluster size*
1L	-24, -98, 16	5.30	0.000	1
1R	36, -88, 18	5.29	0.000	1
2R	28, -78, 24	3.11	0.077	1
3R	34, -90, 20	4.67	0.001	47
4L	-10,-104,16	3.77	0.02	2
4R	36,-90, 16	5.99	0.000	1
5L	-22, -98, 18	7.21	0.000	52
5R	34, -94, 12	5.63	0.000	9

Table 2. Stereo sensitive region: V3B

Subject	*Location*	*Z-Score*	*P corrected*	*Cluster size*
1R	42, -82, 4	(Inf)	0.000	149
2R	38,-90, 2	3.96	0.056	39
3R	40, -80, 0	4.58	0.001	21
4L	-26, -100, 4	7.59	0.000	28
4R	36, -94, 6	(Inf)	0.000	103
5L	-36, -92,4	5.47	0.000	7
5R	36, -88, -4	6.55	0.000	21

Table 3. Stereo sensitive region: Precuneus (BA7)

Subject	*Location*	*Z-Score*	*P corrected*	*Cluster size*
1R	32, -72, 52	7.06	0.000	1
2L	-26, -74, 46	4.67	0.003	4
2R	28, -66, 54	6.01	0.000	96
3R	14, -78, 52	3.90	0.015	5
4R	28, -64, 54	4.30	0.003	8
5R	14, -74, 54	5.47	0.000	1

Table 4. Stereo sensitive region: V1

Subject	*Location*	*Z-Score*	*P corrected*	*Cluster size*
1L	-8, -102, -2	(Inf)	0.000	236
1R	16, -100, 0	(Inf)	0.000	149
2L	-8, -106, -2	6.20	0.000	59
3L	-6,-104,-6	5.27	0.000	55
4L	-14, 106, 0	5.83	0.000	15
4R	16, -100, 6	6.65	0.000	214
5R	10, -100, 4	5.06	0.001	1

5 Discussion

The results of our experiments reveal four main regions sensitive to stereoscopic depth: V1, V3A, V3B and precuneus. In contrast to the evidence from physiological experiments in monkey [2, 19] we did not find evidence that V5 was involved in the processing of stereoscopic information. The activation in V1 observed in experiment 2 is unlikely to be due to the slight differences in the illumination between the different conditions as this was similar in both experiments. It maybe due the difference in disparities used in the two experiments. In Experiment 1 the disparity was 0.3deg and in Experiment 2 it was smaller (±0.076 deg). Because V1 is sensitive to a narrow range of near zero disparities [10, 20] this may account for this difference in the results. Consistent with the results of other studies, we find in both experiments 1 and 2 that V3A and precuneus showed sensitivity to stereo disparities [6, 9, 10, 21]. It is unlikely that these activations are due either to different patterns of eye movements or to stronger attentional engagement. In Experiment 1 the control condition provided identical requirements for these parameters and similar areas of activation were found in Experiment 2 in which no eye movements were required to perform the task.

To our knowledge only one other study has reported the V3B region as being sensitive to stereoscopic information [10]. Other fMRI studies have implicated V3B in motion processing. Orban et al [22-23] showed that this area is sensitive to kinetic boundaries (boundaries defined by motion differences) and Smith and Greenlee et.al showed it to be activated by second order motion. They use the term first order motion to refer to the point to point changes in position over time idealised as a single point of light moving through space. Second order motion is used to refer to the perception of motion arising not from point to point temporal correspondence of luminance, but of correspondence of higher order global properties, which in their study were produced by the contrast modulation of a static texture.

Table 5. Comparative table of functional and anatomical profiles reported for the V3B region. The following tables compare the functional profiles and the average Talaraich coordinates of the V3B region found in the present experiments with those reported by other authors. The standard deviation from the mean is shown in brackets. E1 and E2 refers to the first and second experiments reported in this paper

Ref.	*x*	*y*	*z*	*Sensitive to*
1	-25	-88	-1	Kinetic boundaries.
2	-28	-94	-4	Kinetic contours, shape
	34	-88	0	and motion.
3	±31	-91	0	Kinetic boundaries.
4	±26 (8)	-89 (8)	-2 (8)	Second order motion.
5	*Not given*	*by the*	*author*	Stereopsis.
E1	-27.3 (1.1)	-98.6 (3.1)	0 (5.5)	Stereopsis.
	36.2 (3.9)	-90.75 (5.9)	-0.7 (3.5)	
E2	-31 (7.1)	-96 (5.6)	4 (0)	Stereopsis.
	38.4 (2.6)	-86.8 (5.7)	1.6 (3.8)	

A possible explanation for the activation of V3B in our experiments is that it is due to motion cues introduced by moving shapes defined by coherent disparity information. In Experiment 1 the task required the tracking of a shape over time and in Experiment 2 rotary motion occurred as orientation changed once a second. We propose that activation in the V3B region might be involved in the segmentation process in three dimensional space, required by the global stereo tracking task and the shape discrimination task. Other fMRI studies [21] investigating object shape perception have found that cortical regions (Lateral Occipital) close to V3B were involved in the analysis of object structure independent of the cues (luminance, colour, depth) that define the shape. Other than this we have no explanation as to why the Backus study which used a static depth discrimination task reported V3B as sensitive to stereo disparities. That study did not report the anatomical co-ordinates of the V3B region activated, so we were not able to compare its location with the V3B region reported here. Table 5 shows the functional and anatomical profiles reported for the V3B region.

6 Conclusions

Our results suggest a network involving the cortical areas V1, V3A, V3B and precuneus in the processing of stereo disparities. The high proportion of activations located in the right hemisphere supports the notion of right cerebral dominance in stereo vision. Contrary to the results reported in studies with monkeys, our experiments did not reveal any evidence of the sensitivity of V5 to stereo disparity processing. Our results also showed a region in with functional profile and anatomical location which matched the V3B region. We have suggested that this region was activated by stereoscopic motion component of the dynamic tasks used. Although both tasks used in the experiments (global stereo tracking and the form discrimination) produced motion of spatio-temporal changes of shapes defined by disparity, neither of them were designed specifically to optimally produce disparity based second order motion. The objective of our studies was to process functional magnetic resonance images to investigate the cortical areas selective to stereoscopic information, and an experiment designed to test the hypothesis of the selectivity of the V3B region to stereoscopic motion is for future work.

References

1. Gumming, B.G. and A.J. Parker, Binocular neurons in V1 of awake monkeys are selective for absolute, not relative, disparity. J Neurosci, 1999. 19(13): p. 5602-18.
2. DeAngelis, G.C. and W.T. Newsome, Organization of disparity-selective neurons in macaque area MT. J Neurosci, 1999. 19(4): p. 1398-415.
3. Tootell, R.B., et al., Functional analysis of V3A and related areas in human visual cortex. J Neurosci, 1997. 17(18): p. 7060-78.
4. Braddick, O.J., et al., Brain areas sensitive to coherent visual motion. Perception, 2001. 30(1): p. 61-72.
5. McKeefry, D.J., et al., The activity in human areas V1/V2, V3, and V5 during the perception of coherent and incoherent motion. Neuroimage, 1997. 5(1): p. 1-12.
6. Gulyas, B. and P.E. Roland, Binocular disparity discrimination in human cerebral cortex: Functional anatomy by positron emission tomography. Proc. Natl. Acad. Sci. USA (Nuerobiology), 1993. 91: p. 1239-1243.
7. Kwee, I.L., et al., Perceptual processing of stereopsis in humans: high-field (3.0-tesla) functional MRI study. Neurology, 1999. 53(7): p. 1599-601.

8. Hanazawa, A., et al., The human posterior parietal cortex participates in stereoscopic depth perception. A fMRI study. NeuroImage (poster), 2000. 11(5).
9. Nishida, Y., et al., Stereopsis-processing regions in the human parieto-occipital cortex. Neuroreport, 2001. 12(10): p. 2259-63.
10. Backus, B.T., et al., Human cortical activity correlates with stereoscopic depth perception. J Neurophysiol, 2001. 86(4): p. 2054-68.
11. Freeman, R.D., Stereoscopic vision: Which parts of the brain are involved? Curr Biol, 1999. 9(16): p. R610-3.
12. Pelli, D.G., The videotoolbox software for visual psychophysics: Transforming numbers into movies. Spatial vision, 1997. 10: p. 437-442.
13. Brainard, D.H., The psychophysics toolbox. Spatial vision, 1997. 10: p. 433-436.
14. Friston, K.J., et al., Statistical parametric maps in functional imaging: A general linear approach. Human Brain Mapping., 1995. 2: p. 189-210.
15. Worsley, K.J., et al., A three-dimensional statistical analysis for CBF activation studies in human brain. J Cereb Blood Flow Metab, 1992. 12(6): p. 900-18.
16. Friston, K.J., A.P. Holmes, and K.J. Worsley, How many subjects constitute a study? Neuroimage, 1999. 10(1): p. 1-5.
17. Corbetta, M., et al., A common network of functional areas for attention and eye movements. Neuron, 1998. 21(4): p. 761-73.
18. Petit, L. and J.V. Haxby, Functional anatomy of pursuit eye movements in humans as revealed by fMRI. J Neurophysiol, 1999. 82(1): p. 463-71.
19. Gumming, B.C. and G.C. DeAngelis, The physiology of stereopsis. Annu Rev Neurosci, 2001. 24: p. 203-38.
20. Poggio, G.F., F. Gonzalez, and F. Krause, Stereoscopic mechanisms in monkey visual cortex: binocular correlation and disparity selectivity. J Neurosci, 1988. 8(12): p. 4531-50.
21. Mendola, J.D., et al., The representation of illusory and real contours in human cortical visual areas revealed by functional magnetic resonance imaging. J Neurosci, 1999. 19(19): p. 8560-72.
22. Dupont, P., et al., The kinetic occipital region in human visual cortex. Cereb Cortex, 1997. 7(3): p. 283-92.
23. Orban, G.A., et al., A motion area in human visual cortex. Proc Natl Acad Sci USA, 1995. 92(4): p. 993-7.

An Image Analysis System to Compute the Predominant Direction of Motion in a Foucault Pendulum

Joaquín Salas[1] and Jésica Flores[2]

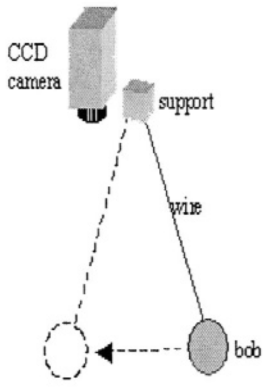
[1] Institute Politécnico Nacional*
salas@ieee.org
[2] Instituto Tecnológico de Querétaro

Abstract. In this document, we propose a machine vision system to detect the predominant direction of motion of a Foucault pendulum. Given a certain configuration where the camera has a top view of the pendulum's bob in motion, the system builds an adaptive model of the background. From it, the bob's center of mass is computed. Then, an ellipse model is fitted to the trajectory. Finally, the noise in the observed predominant direction of motion is filtered out to get a robust estimate of its value. The system has proved to be quite reliable on a simple version of the Foucault pendulum where it was tested.

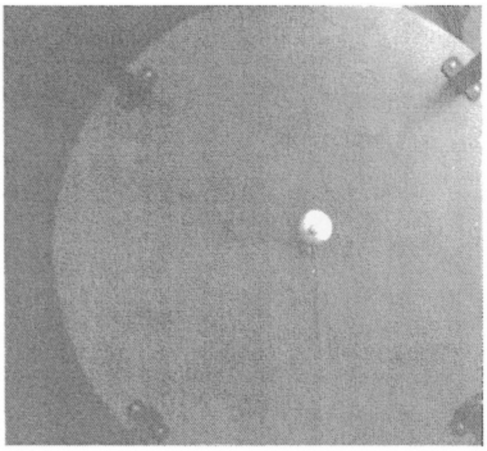
1 Introduction

A Foucault pendulum is an important device for a number of scientific and educational reasons which include: a) It can be used to show that the Earth spins; b) It can be used to provide accurate time-keeping; and, c) It can be used to measure g, the acceleration due to gravity. In 1851, French physicist Jean Bernard Leon Foucault proved that the Earth rotates by swinging a pendulum attached to a building[9]. Foucault observed the forces acting on the pendulum. Since there is not force making the pendulum rotate with respect to the floor, Foucault concluded that it must be the floor that rotates with respect to the pendulum. Currently, observations on the pendulum are done without the aid of automatic instruments. This is wearisome and error prone. To appreciate what it takes, consider Gillies' review[6] of Maurice Allais observations of the Foucault pendulum during an eclipse in 1954. Guilles reports that in his experiment, Maurice Allais observed the pendulum every 14 minutes during 30 days and nights without missing a data point. In this paper, we propose a machine vision system to make these observations, and from them to study physical phenomenas. For example, during major eclipses around the world[6,1,5], it has been observed

* We would like to express our thanks to Prof. Reynaldo Pless for providing us with his Foucault pendulum. This work was partially funded with a grant from CEGEPI-IPN.



(a) A bob, hanging from a support through a wire, swings back and forth. A CCD camera is placed close to the gripping system.



(b) Camera's top view of the experimental setup.

Fig. 1. A machine vision system to compute the predominant direction of motion in a Foucault pendulum

how implementations of the pendulum experimented a perturbation by describing an ellipse whose major axis deviated in relation to the predicted motion plane.

Current state of the art implementations of a Foucault pendulum are aimed to beat problems such as the precession due to ellipticity[2]; and, the loss of amplitude due to the gripping method used to hold the wire to the building[10]. With these features, a Foucault pendulum can be used to keep track of time. For a pendulum in the north/south pole the floor under the pendulum would twist around the Earth's axis every 24 hours. For a pendulum on the equator the floor would not twist at all but the building would travel eastward on the Earth's axis. For places at different latitude some twisting and some travelling takes place. However, while the twisting can be seen the traveling motion of the pendulum eastwards can not. The degree of twist depends on the latitude ϕ by $n = 360^o \sin\phi$[9]. For instance, consider a point in Mexico city's with latitude $19^o00'26''$N. There, the pendulum will rotate about $117^o14'50''$ in 24 hours.

In this document, we present a vision system to measure the direction of motion of a Foucault pendulum. To that end, we grab images from a camera placed close to the gripping system, aligned with the vertical direction (see Fig. 1). Our system tackles three different aspects of the problem. On the one hand, in §2, we describe how the bob can be detected under changes in the background scenario. Then, in §3, we compute the bob's trajectory by fitting an ellipse to the set of observed positions. Finally, in §4, the noise in the estimated predominant direction of motion is filtered out using a Kalman filter.

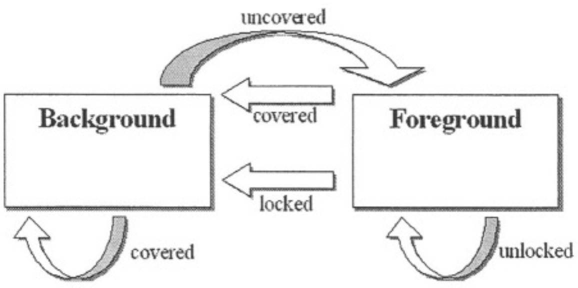


Fig. 2. State diagram for detecting the Foucault pendulum bob (see the text for details)

2 Bob Detection

Background subtraction is a technique commonly used to detect moving objects in a sequence of images. It has the advantage that it is easy to compute. However, due to dynamic changes in the scene being observed, in most cases an adaptive background construction model needs to be implemented for long term observation. Nowadays, a number of adaptive techniques have been proposed. Current solution models show a clear trend. Most of them model the pixel's intensity value dynamics with a set of statistical models. For instance, Stauffer and Grimson[12] use a mixture of gaussian models. For a given observation, the most suitable model is used. Monnet *et al.*[8] use Principal Component Analysis to create a model of the variability of the pixel dynamics over a sequence of frames. The principal components are computed on the covariance matrix built from the pixel intensity, over the previous m observations. Others, like Davis *et al.*[3], make no prior assumption about the underlying statistical model until enough evidence is accumulated. It has been argued that Kalman-based approaches[11] are robust but respond slowly to changes. For this work, we implemented an adaptive background construction model based on several Kalman filters for a given pixel. A particular filter is used depending on how well it describes the current pixel process status. Otherwise, a new filter is initiated. The model has the advantage of responding quickly to changes in illumination conditions because it retains a certain extent of memory about historic background tendencies.

Kalman filters address the problem of estimating the state $\mathbf{x} \in \mathcal{R}^n$ of a discrete time controlled process[7]. The dynamics of a linear stochastic process can be described for iteration k by the linear stochastic differential equation [14]

$$\hat{\mathbf{x}}_{k+1|k} = A_k \hat{\mathbf{x}}_{k|k} + B_k \mathbf{u}_k \tag{1}$$

with a measurement $\mathbf{z}_k = H_k \mathbf{x}_k$. Here $\mathbf{u}$ is the input, and $\mathbf{x}$ is the state; A is the state propagation matrix, B is the input matrix, and H_k is the output matrix. The *a priori* estimated error covariance is given by[13]

$$P_{k+1|k} = A_k P_{k|k} A_k^T + Q_k \tag{2}$$

where Q_k is the process noise covariance. At time k, two pieces of data are available. One is the estimate $\hat{\mathbf{x}}_{k|k-1}$ of the state $\mathbf{x}_k$ given measurements up to but not including $\mathbf{z}_k$.

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + K_k(\mathbf{z}_k - H_k\hat{\mathbf{x}}_{k|k-1}) \tag{3}$$

The error covariance matrix $P_{k|k}$ is predicted from the previous observations $P_{k|k-1}$ as $P_{k|k} = (I - K_kH_k)P_{k|k-1}$. The second piece of data is the gain or blending factor K_k that minimizes the *a posteriori* error covariance and that is given by

$$K_k = P_{k|k}H_k^T(H_kP_{k|k}H_k^T + R_k)^{-1} \tag{4}$$

where R_k is the measurement noise covariance.

Up to two filters, $F = \{f_1, f_2\}$, are defined for each image pixel p. How many and which one is used depends on how well the pixel dynamics is modeled by the filter (see Fig. 2). At a particular point in time a pixel is either part of the background or the foreground. For a given filter, it is said to be *locked* when its error covariance $P^j_{k|k}$ converges. Otherwise, it is said to be *unlocked.* A pixel is said to be covered when its intensity value is within a certain number α of error covariances $P_{k|k}$ from the current state estimate value $\mathbf{x}_k$. Initially, all pixels are part of the foreground. As the time pass by, most of the filters are *locked.* Then the pixels they are included into become part of the background. Some of the pixels remain *unlocked.* They keep being part of the foreground. When a pixel is part of the background, it may be that a pixel intensity value is not covered. This will start a new filter process and will file the pixel as being part of the foreground. This is the typical case of an object passing in front of a static object. When the moving object finally pass by the previous filter is still covering the intensity values and comes back to continue the process.

3 Trajectory Description

In general, due to loss of energy, the top view trajectory of the bob is described by an ellipse. The general form of a quadratic curve is given by

$$F(\mathbf{a}, \mathbf{x}) = \mathbf{a}^T\mathbf{x} = ax^2 + bxy + cy^2 + dx + ey + f = 0 \tag{5}$$

where $\mathbf{a}^T = (a, b, c, d, e, f)$, $\mathbf{x} = [x^2, xy, y^2, x, y, 1]$, and (x, y) is a point in the image plane. The constraint for Eq. (5) to represent an ellipse is $b^2 < 4ac$. Fitzgibbon *et al*[4], proposed a method to look for the ellipse's parameters through the solution to the generalized eigenvalue problem of the system

$$S\mathbf{u} = \lambda C\mathbf{u} \tag{6}$$

where $S = D^TD$ is a scattered matrix and C express the constraint. That is, $D = (\mathbf{x}_1, \ldots, \mathbf{x}_n)^T$ is the design matrix and C is a 6×6 matrix with all entries zero but $C(3, 1) = C(1, 3) = 2$ and $C(2, 2) = -1$. Fitzgibbon *et al.* show that the solution to the system in Eq. (6) is always an ellipse for the eigenvector corresponding to the only negative eigenvalue. Furthermore, Eq. (5) can be transformed into

$$F(\mathbf{a}',\mathbf{x}') = \mathbf{a}'^T\mathbf{x}' = a'x'^2 + b'x'y' + c'y'^2 + d'x' + e'y' + f' = 0 \quad (7)$$

by virtue of the rotation $\mathbf{x}' = R\mathbf{x}$, with $R = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$. The relationship between $\mathbf{a}$ and $\mathbf{a}'$ is

$$\begin{aligned} a' &= a\cos^2\theta + b\sin\theta\cos\theta + c\sin^2\theta \\ b' &= 2(c-a)\sin\theta\cos\theta + b(\cos^2\theta - \sin^2\theta) \\ c' &= a\sin^2\theta - b\sin\theta\cos\theta + c\cos^2\theta \\ d' &= d\cos\theta + e\sin\theta \\ e' &= e\cos\theta - d\sin\theta \\ f' &= f \end{aligned} \quad (8)$$

There is an angle θ for which the term b' disappears. This is when the ellipse's major axes are parallel to the reference system principal axes. In this case, the angle θ is given by

$$\theta = \begin{cases} \frac{1}{2}\arctan\left(\frac{b}{a-c}\right) & \text{for } a \neq c \\ 45^o & \text{otherwise} \end{cases} \quad (9)$$

The ellipse's center can be found at $\mathbf{a} = \mathbf{a}'' + (h,k)^T$. Replacing this value into Eq. (5) gives

$$\begin{aligned} ax''^2 + bx''y'' + cy''^2 + (2ah + bk + d)x'' + (bh + 2ck + e)y'' + \\ (ah^2 + bhk + ck^2 + dh + ek + f) = 0 \end{aligned} \quad (10)$$

For the ellipse to be centered at the origin both coefficients getting along with x'' and y'' have to go to zero. This leads to the system

$$\begin{pmatrix} 2a & b \\ b & 2c \end{pmatrix} \begin{pmatrix} h \\ k \end{pmatrix} = - \begin{pmatrix} d \\ e \end{pmatrix} \quad (11)$$

This way EC. (9) and (11) give us both the orientation and the center of the ellipse, respectively.

4 Direction of Motion

Earth rotation frequency ω_e has a vertical and a northward pointing horizontal component of magnitude $\omega_e' = \omega_e \sin(l)$ and $\omega_e'' = \omega_e \cos(l)$ respectively, where l is a particular place on Earth latitude. The vertical component causes the Foucault pendulum to precess clockwise at a ω_e'' rate when viewed from above. The horizontal component can be observed as the rate of the passing stars viewed directly overhead. Latitude l is the angular distance from the equator. The pendulum is expected to precess due to Earth rotation by

$$\theta(t) = \omega_e'' t \quad (12)$$

That is, the angle depends lineally on time.

Kalman filtering is a procedure to robustly estimate iteratively state parameters from a series of data readings. The filter combines optimally the data readings and the current state estimation to produce, based on the uncertainty about both, a better estimation of the state. The filter predicts based on the accumulated evidence. New readings are used to update the values. Given the general framework for state estimation by Kalman filtering, the initial state of the bob is $\hat{\mathbf{x}}_0 = \left(\dot{\theta}, \theta\right)^T$. The orientation at time t is given by $\theta(t) = \theta(0) + \dot{\theta}(0)t$. This is a continuous equation. Since, we are taking measurements at discrete time intervals, this relation becomes

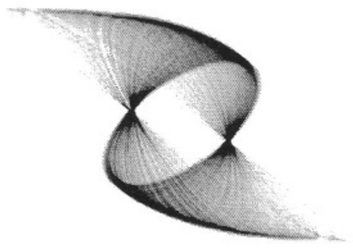
$$\theta_{k+1} = \theta_k + \dot{\theta}_k dt \tag{13}$$

Thus, one can write the system update equation as $\mathbf{x}_{k+1} = F\mathbf{x}_k$, where $\mathbf{x}_k = \left(\hat{\theta}_k, \theta_k\right)^T$ and $F = \begin{pmatrix} 1 & 0 \\ dt & 1 \end{pmatrix}$.

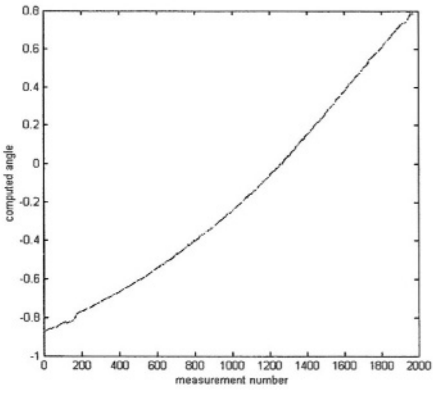
5 Experimental Results

We tested the mathematical framework exposed in the previous sections. In particular, we use a pendulum with flexible steel cable of length 1,16m with a 0,200kg iron bob. The wire is attached to an iron collar. This simple system is subject to severe effects due to friction with the collar. Nevertheless, this platform provide us with an experimental setup to prove our algorithms. The time required for the pendulum to describe an ellipse is called its period. The formula to calculate this quantity is $t = 2\pi\sqrt{\frac{l}{g}}$, where l is the pendulum length in meters and g is the gravitational field strength. This quantity at sea level is about 9.81 $\mathrm{m/sec}^2$. In our case, it means that our pendulum will turn around in about 2.1606sec.

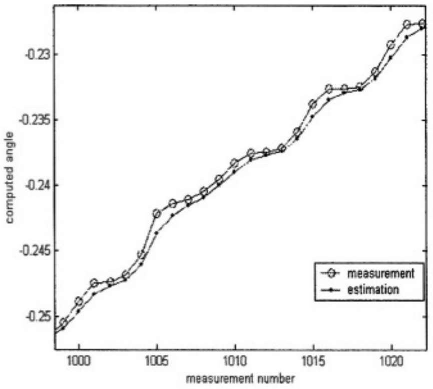
The adaptive background construction model introduced in §2 allows us to build a good approximation to the scene static objects. This model adapts well to changes in illumination conditions and different scenarios. This is important because it allows a more ample set of environments where the vision system can be deployed. Error, process and measurement covariance errors were given initial values 16.0, 0.1 and 49.0 respectively. The pendulum center of mass was computed from the resulting foreground image. For the ellipse fitting algorithm, we used the last 20 observed positions of the center of mass. Then the Matlab routine for the generalized eigenvalue problem was incorporated in our Visual C++ implementation. After the first 20 frames, for every new center of mass value, an ellipse is computed. For each new value of the center of mass, we discard the older one using a *First In, First Out* (FIFO) policy. For an experimental run with 2,000 images, the results are shown in Fig. 3. Fig. 3(a) shows the resulting elliptic trajectories estimated. The ellipse's major axis is selected as the predominant direction of motion. Its orientation for this run is presented in Fig. 3(b) and 3(c).



(a) Drawing of the elliptic trajectory described by the bob during the experiment



(b) Computed orientation



(c) Detail in (b)

Fig. 3. Computing the predominant direction of motion. The sequence has 2000 frames. The detail shows about 20 observations

6 Conclusion

In this document, we described a complete image analysis system to compute the predominant direction of motion of a Foucault pendulum. The system can be used in a wide range of scenarios since it builds an adaptive model of the background. At the same time, the systems constructs a description of the bob's trajectory from the viewpoint of the pendulum's support. Finally, the observed angular orientation of the ellipse's major axis is filtered using a Kalman formulation. As a whole, the system has proved to be a robust, accurate and complete solution to the problem.

Our experimentation with a simple and brittle system suggests that our implementation will behave just as well or even better with a more robust construction of Foucault pendulum.

References

1. William R. Corliss. *Science Frontiers: Some Anomalies and Curiosities of Nature.* Sourcebook Project, 1994.
2. Richard H. Crane. A Foucault Pendulum Wall Clock. *American Journal of Physics,* 63(1):33–39, 1995.
3. A. Elgammal, R. Duraiswami, D. Harwood, and L.S. Davis. Background and Foreground Modeling using Nonparametric Kernel Density Estimation for Visual Surveillance. *Proceedings of the IEEE,* 90(7):1151 – 1163, 2002.
4. A. Fitzgibbon, M. Pilu, and R.B. Fisher. Direct Least Square Fitting of Ellipses. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 21(5):476 – 480, 1999.
5. Amitabha Ghosh and Soumitro Banerjee. A proposed Observation During the Total Solar Eclipse on 11 August 1999. *eprint arXiv:astro-ph/9904083,* page 1999astro.ph..4083G, 1999.
6. G. T. Gillies. Difusse Reflection. *American Journal of Physics,* 58:530, 1990.
7. R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal of the Basic Engineering,* 82:35–45, 1960.
8. A. Monnet, A. Mittal, N. Paragios, and Visvanathan Ramesh. Background Modeling and Subtraction of Dynamic Scenes. In *IEEE International Conference on Computer Vision,* volume 2, pages 1305 – 1312, 2003.
9. The California Academy of Sciences. http://www.calacademy.org/products/pendulum.html, 2004.
10. Andreas Rasche, Peter Troger, Michael Dirska, and Andreas Poize. Foucault's Pendulum in the Distributed Control Lab. In *IEEE International Workshop on Object-Oriented Real-Time Dependable Systems,* pages 299–306, 2003.
11. G. Rigoll, S. Muller, and B. Winterstein. Robust Person Tracking with Non-Stationary Background using a Combined Pseudo-2D-HMM and Kalman-Filter Approach. In *IEEE International Conference on Image Processing,* volume 4, pages 242–246, 1999.
12. Chris Stauffer and W. E. L. Grimson. Adaptative Background Mixture Models for Real-Time Tracking. In *IEEE Conference on Computer Vision and Pattern Recognition,* pages 246–252, 1999.
13. Carlo Tomasi. *Mathematical Methods for Robotics and Vision.* CS205 Stanford University, 1994.
14. Greg Welch and Gary Bishop. An Introduction to the Kalman Filter. Technical report, University of North Carolina, 2004.

A Perceptual User Interface Using Mean Shift

Edson Prestes[1], Anderson P. Ferrugem[1], Marco A. P. Idiart[2], and Dante A. C. Barone[1]

Universidade Federal do Rio Grande do Sul,
[1] Institute de Informática-UFRGS, P.O. Box 15064, and
[2] Institute de Física-UFRGS, P.O. Box 15051,
91501-970 Porto Alegre, RS - Brazil
{prestes, ferrugem, idiart, barone}@inf.ufrgs.br

Abstract. This work describes a perceptual user interface based on the Mean Shift algorithm to control mouse events by gestures using a generic usb camera. To demonstrate the usefulness of our work, we present two preliminaries experiments that are controlled by the mouth. The first experiment corresponds to a specific application in charge of controlling a game called pitfall. The second one is a generic experiment that evaluates the precision and robustness of the interface. These preliminaries results show that potentiality and applicability of our study to disable people.

1 Introduction

The body expressions are natural ways used by humans and animals to communicate, express feelings and internal intentions in the society. During the last years, they have been used to build interfaces more intuitive than the traditional ones based on keyboard or mouse. These interfaces, commonly called *perceptual user interface*(PUI) gave rise to a new concept of interaction between man and machine.

The PUIs allow the user to have a bidirectional interaction with the computer in a simplified way. They are good candidates in learning, monitoring and accessibility tasks, such as teaching of deaf sign language, studying of athlete performance, helping disable people to use the computer, commercial computer games, immersive 3D world, etc.

Such new generation of interfaces will improve substantially the execution of tasks through the parallelization of user actions [1]. For example, the use of vision as a second stream of input, in addition to mouse, allows the interface to perceive the user, classify his movements and activities by reacting accordingly [2]. In short, they are good hand-free alternative and/or extension to conventional pointing devices. Furthermore, they are cheaper as compared to early systems that required expensive dedicated hardware like headgear or data glove.

Several interfaces and techniques have been developed during the last years, for instance, Toyama [3] proposed the use of head motion to position the cursor in a GUI through the Incremental Focus of Attention; Li [4] developed an interface that is able to lip-reading using eigensequences; Bérard[1] proposed a technique that uses head motion to navigate in a document; Davis [5] developed a PUI that uses PupilCam together with anthropometric head and face measures to recognize user acknowledgments from head gestures. Nishikawa [6] developed an interface based on vision for controlling the position of a laparoscope and so on.

In this paper, we propose a perceptual interface based on vision that uses as core the Mean Shift algorithm [7, 8]. This algorithm is fast and efficient being an excellent candidate to real time task. It has been used successfully in task of tracking of objects during the last years. In our interface, it is used to track and to interpret gestures.

This paper is organized as follows. The Section 2 presents the Mean Shift algorithm. The Section 3 shows the Perceptual Interface. The Section 4 shows the experiments and, finally, the Section 5 presents the future works and conclusions.

2 Mean Shift

The Mean Shift Algorithm was proposed by Comaniciu [8, 9] to track non-rigid objects with different colors and texture patterns in real-time. The basic idea is to persue a desired target in the current frame with a fixed-shape window. The algorithm receives as input model a target color distribution and monitors a candidate region whose color content matches the input model. It estimates the position of the target in the next image frame where supposedly the difference among the color distribution of the candidate region and the input model is smaller than a predefined threshold[8]. This difference is calculated from the Bhattacharyya coefficient [10] that provides a trusty similarity measure.

This method has proved to be robust to partial occlusions of the target, large variation in the target scale and appearance, rotation in depth and changes in camera position [8]. Furthermore, Comaniciu [9] showed that spatially masking the target with an isotropic kernel permits the use of a gradient optimization method to perform an efficient target localization compared to exhaustive search methods.

The algorithm works as follows. Initially, it receives as input the *color model* of the target represented as a probability density function (*pdf*) in the feature space; secondly, it estimates the new target position through a mean vector calculated from Mean Shift Algorithm. Next, the candidate region move to the direction pointed by the mean vector, and the process proceed.

2.1 The Target Model

The aim of the method is to follow a given object or feature that moves in a scene. The first step, therefore, is to characterize such image. A model of the object is chosen in the following way: A circular region, of radius h, centered in the object's center $\mathbf{x_c}$ and encompassing it totally is selected. For each point (or pixel) $\mathbf{x} = (x_1, x_2)$ in the region a feature vector is extracted and categorized according to a discrete number of prototypical features, and the point receives the index of that feature, $u = b(\mathbf{x})$. The feature distribution $\mathbf{q} = \{q_u\}_{u=1\ldots m}$, that accounts for the fractional occurrence of a given feature u in the object's region, is calculated by

$$q_u = \frac{\sum_{i=1}^{n} \kappa(|\mathbf{x}_i - \mathbf{x}_c|/h)\, \delta(b(\mathbf{x}_i), u)}{\sum_{i=1}^{n} \kappa(|\mathbf{x}_i - \mathbf{x}_c|/h)}. \tag{1}$$

where, $\mathbf{x}_c$ is the center of the region and δ is the Kronecker delta function[1]. Observe that the distribution satisfy $\sum_{u=1}^{m} q_u = 1$.

The function $\kappa(x)$ is an isotropic kernel with a convex and monotonic decreasing profile. Its goal is to reduce the importance of the peripheral features when calculating the target's $\mathbf{q}$ distribution. In our experiments, we choose the Epanechnikov Kernel

$$\kappa(x) = \begin{cases} \frac{1}{2} C_d^{-1}(d+2)(1-x^2) & \text{if } x \leq 1 \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

where C_d is the volume of the unit d-dimensional sphere [11], in our case $d = 2$, and $C_d = 4/3\,\pi$.

Specifically, the important features are color. The $\mathbf{q}$ distribution represents a color histogram $\mathbf{q} = \{q_u\}_{u=1\dots m}$ that incorporates color and spatial information of the image pixels.

2.2 The Candidate Region

A candidate region is a region whose feature distribution is highly similar to the target color model. In a given image frame we define the feature distribution for a given point $\mathbf{y}$ in the image,and scale h_e, as

$$p_u(\mathbf{y}, h_e) = \frac{\sum\limits_{i=1}^{n} \kappa(|\mathbf{x}_i - \mathbf{y}|/h_e)\ \delta(b(\mathbf{x}_i), u)}{\sum\limits_{i=1}^{n} \kappa(|\mathbf{x}_i - \mathbf{y}|/h_e)}. \tag{3}$$

The scale h_e defines a tentative size for the object's circular region.

The similarity of the two distributions can be calculated by the Bhattacharyya coefficient

$$\rho(\mathbf{y}) = \rho[\mathbf{p}(\mathbf{y}), \mathbf{q}] = \sum_{u=1}^{m} \sqrt{p_u(\mathbf{y})}\,\sqrt{q_u} \tag{4}$$

This coefficient can be viewed geometrically as the cosine of the angle between the m-dimensional unit vector $p = (\sqrt{p_1}, \dots, \sqrt{p_m})$ and $q = (\sqrt{q_1}, \dots, \sqrt{q_m})$.

2.3 Target Localization

The target is localized at the point $\mathbf{y}^*$, and scale h^*, such that $\rho(\mathbf{y}^*, h^*) = \max_{y,h} \rho(\mathbf{y}, h)$, i.e., the candidate region with the highest similarity.

For real time applications an exhaustive search in $(\mathbf{y}, h)$ space is hopeless, therefore we have to adopt an incremental procedure where the candidate region suffers small corrections at each image frame. It is a reasonable approximation when the object's motion is small during the time between frames.

The adequate method is to adopt the gradient ascent on the similarity function, or

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \eta \nabla \rho(\mathbf{y}_n) \tag{5}$$

[1] The Kronecker delta function returns 1 if its arguments are equal and 0, otherwise.

Here we focus to the case where the scale $h_e = h$ is fixed at the correct one, and the only the position $\mathbf{y}$ is adjusted. Considering that $\mathbf{y}_0$ is the current estimate of the object's position, the similarity function can be expanded around this value for small corrections using that

$$p_u(\mathbf{y}, h) = p_u(\mathbf{y}_0, h) + (\mathbf{y} - \mathbf{y}_0) \cdot \nabla p_u|_{y=y_0} \ ,$$

as

$$\rho(\mathbf{y}) \approx \frac{1}{2}\sum_{u=1}^{m} \sqrt{p_u(\mathbf{y}_0) q_u} + \frac{1}{2}\sum_{u=1}^{m} p_u(\mathbf{y}) \sqrt{\frac{q_u}{p_u(\mathbf{y}_0)}} \tag{6}$$

Introducing (3) in (6), we obtain

$$\rho(\mathbf{y}) \approx \frac{1}{2}\rho(\mathbf{y}_0) + \frac{1}{2}\frac{\sum_{i=1}^{n} w(\mathbf{x}_i)\ \kappa\left(|\mathbf{y} - \mathbf{x}_i|/h\right)}{\sum_{i=1}^{n} \kappa\left(|\mathbf{y} - \mathbf{x}_i|/h\right)} \tag{7}$$

where the weight $w(\mathbf{x}_i)$ is given by

$$w(\mathbf{x}_i) = \sum_{u=1}^{m} \delta(b(x_i), u) \sqrt{\frac{q_u}{p_u(\mathbf{y}_0)}} = \sqrt{\frac{q_{b(\mathbf{x}_i)}}{p_{b(\mathbf{x}_i)}(\mathbf{y}_0)}} \tag{8}$$

The ratio $r_u = q_u / p_u(\mathbf{y}_0)$ indicates if the color u, for the current estimation of the object's position $(\mathbf{y}_0)$, is above $(r_u < 1)$ or below $(r_u > 1)$ the model's prediction. Therefore the weight $w(\mathbf{x}_i)$ indicates the importance the pixel $\mathbf{x}_i$ has in correcting the object's color distribution. After some manipulation, the gradient of the eq. (7) can be written as

$$\Delta\mathbf{y} = \frac{\sum_{i=1}^{n} (\mathbf{x}_i - \mathbf{y}_0)\ w(\mathbf{x}_i)}{\sum_{i=1}^{n} w(\mathbf{x}_i)} \tag{9}$$

where it was used that derivative of the Epanechnikov kernel is linear in position. Equation (9) is also called the Mean Shift vector. It points to the direction the center of the kernel has to move in order to maximize the similarity between the colors distributions $\mathbf{p}(\mathbf{y})$ and $\mathbf{q}$.

The new estimate for the object's position is simply

$$\mathbf{y}_1 = \frac{\sum_{i=1}^{n} \mathbf{x}_i\ w(\mathbf{x}_i)}{\sum_{i=1}^{n} w(\mathbf{x}_i)} \tag{10}$$

It can viewed as the mean position calculated from each pixel position $\mathbf{x}_i$ weighted by the ratio r_u of its color. In the application algorithm if $\mathbf{y}_1$ overshoots, i.e., if $\rho[\mathbf{p}(\mathbf{y}_1), \mathbf{q}] < \rho[\mathbf{p}(\mathbf{y}_0), \mathbf{q}]$, we do $\mathbf{y}_1 \rightarrow \frac{1}{2}(\mathbf{y}_0 + \mathbf{y}_1)$ until some improvement is attained.

3 Perceptual Interface

Our interface uses the meanshift algorithm to track and to interpret gestures using a generic usb camera. Initially, the user selects the target to be tracked, for instance the nose, mouth, hand, etc. Next, he determines actions and associates them to different target configurations. We adopted two different controlling strategies. The first is generic: the user controls the computer using a virtual mouse, i.e., the mouse movement and click events are set by gestures. The mouse movement is done interpreting the target position and the click event is determined by the target configuration. The second strategy is position specific, it depends on the absolute position of the target at the image frame captured by the webcam. Both will be discussed below.

3.1 Target Selection

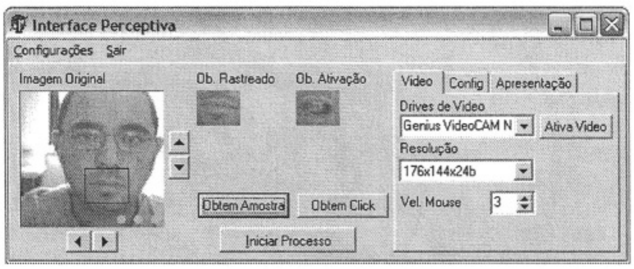
The selection is done by delimiting a region that contains the target in the image coming from the video stream using a box. The Figure 1 shows the selection of a target region to be tracked. In this case, the target corresponds to the mouth of the researcher. Using the same idea, the user can associated different target configurations to mouse actions. For instance, the Figure 1 shows two target configurations. The mouth opened is associated to the left mouse click whereas the mouth closed is associated to the track algorithm to move the pointer mouse in the environment.

3.2 Neutral Zone

Neutral zone corresponds to a specific radial area in the current image frame where no mouse movement or click event is performed. It is defined in the beginning of the control process; its center $\mathbf{c}^z = (c_1^z, c_2^z)$ corresponds to the center of the target and the its radius r_z is set by the user. The neutral zone controls the sensitivity of the system. When the center of the target is outside of the neutral zone, the control takes place. The region 5 in Figure 2 corresponds to the neutral zone.

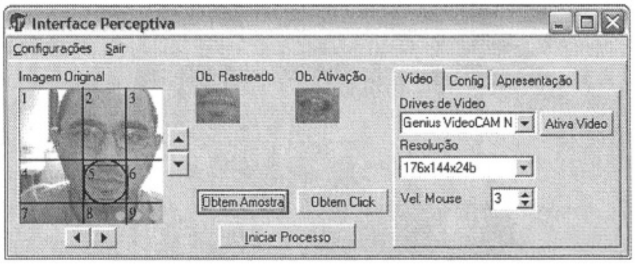
3.3 Specific Control Management

In the specific control, the user maps regions outside of neutral zone into actions. The Figure 2 illustrates eight regions commolly used. In our experiments with Microsoft



Drives de Video: Genius VideoCAM N
Resolução: 176x144x24b
Vel. Mouse: 3

Fig. 1. Target selection. The red box delimits the region to be tracked



Drives de Video: Genius VideoCAM N
Resolução: 176x144x24b
Vel. Mouse: 3

Fig. 2. Image regions used to define different actions

Powerpoint, we used the regions 4 and 6 to go forward and go back in powerpoint presentation, respectively. On the other hand, in our experiments with Windows Media Player, we used the regions 4 and 6 to go forward and back, respectively, in the playlist; and the regions 2 and 8 to increase and decrease volume, respectively. In this case, it was necessary four different regions to control the application.

An action a_n associated to a region n is executed only when a transition from the neutral zone to the region n occurs. No action is executed when a transition between region outside of neutral zone happens. After the execution of an action, the user must return to the neutral zone to activate other action. We intended to implement a mechanism to shoot a sequence of actions instead of a single action each time. In place of performing m transitions from the neutral zone to the region n to execute m times the action a_n, the user is going to need to keep the target a period of time in the region n. In this case, each action is performed at each τ seconds.

3.4 Generic Control Management

In the generic control, the user can move the mouse pointer and shoot the left mouse click event anywhere. The mouse pointer motion is produced from the displacement of the target in the source image. When the center target $\mathbf{c}^t = (c_1^t, c_2^t)$ is outside of the neutral zone,

$$\sqrt{(c_1^t - c_1^z)^2 + (c_2^t - c_2^z)^2} > r_z,$$

the target displacement vector $\mathbf{d} = (d_1, d_2)$ is computed, where $d_1 = c_1^t - c_1^z$ and $d_2 = c_2^t - c_2^z$.

The mouse position is updated as follows. Initially, we calculate the target displacement Δd beyond the neutral zone. This displacement has an upper bound u to constrain the pointer speed, $\Delta d = \min\{|d| - r_z, u\}$. The upper bound value u is defined by the user in run-time.

The speed v is $(\Delta d/u) * v_{max}$, where v_{max} defines the maximum speed of the pointer mouse at each step. The mouse pointer position $\mathbf{p} = (p_1, p_2)$ is updated by

$$\mathbf{p_{t+1}} = \mathbf{p_t} + \left(\frac{d_1}{|d|}, \frac{d_2}{|d|}\right) v \tag{11}$$

This update is performed at each algorithm step in a similar way to a joystick. The Equation 11 controls the mouse speed according to v, i.e., the greater the value v the faster the pointer mouse will move. It allows a smooth, rapid and precise motion in relation to a linear update, where the displacement of the pointer mouse is constant.

To simulate the left mouse event, we need to determine when the target configuration associated to tracking has changed to the target configuration related to the click event. To detect such change, initially, we need to extract the color models of both configurations using the eq.(1).

Consider $\mathbf{q}_t$ and $\mathbf{q}_c$ the color models of the target configurations associated to track and click events, respectively. To determine what action to take, we need to compute the similarity between the candidate region, discussed in Section 2.2, and these color models. This computation aims to identify what model best matches with candidate region, and it is done only when the target center is inside the Neutral Zone.

Misclassification can occur due to luminosity differences. To avoid it, we execute an action only when the target configuration associated is identified more than t times successively, otherwise, the configuration is interpreted as a noise.

4 Experiments

This section presents two experiments using our interface. The first one shows results of the specific control management. In this case, we present some snapshots of the interface controlling a game called *pitfall*. The second experiment is related to generic control management. It aims to provide a quantitative measure of the interface related to learning and motion precision. All experiments use rgb color space. We observed that rgb outperforms hsi, because rgb automatically incorporates spatial information as deepness generally disregarded by hsi color model. This feature allows the track algorithm to follow specific parts of an object with the same color.

4.1 Specific Control Management Experiment

This experiment illustrates the usefulness of the interface in the control of a commercial game using the mouth. This game consist of guiding a treasure hunter in a jungle. The hunter runs and jumps through a 2D sideview environment, avoiding hazards like crocodile-filled waters, sinkholes, etc. The Figure 3 shows the researcher controlling the hunter.

In Figure 3, the blue circle corresponds to the neutral zone and the red box delimits the target. Observe the mouth position, in relation to the neutral zone, determines the action that the hunter should execute. When the mouth is in the regions 4 or 6, the hunter moves to the left or to the right, respectively (see Section 2). When it is in the regions 2 or 8, the hunter jumps or stoops.

4.2 Generic Control Management Experiment

This experiment aims to show the precision and easiness of our interface. We use a maze-like environment, illustrated in Figure 4. The user was in charge of leading the red

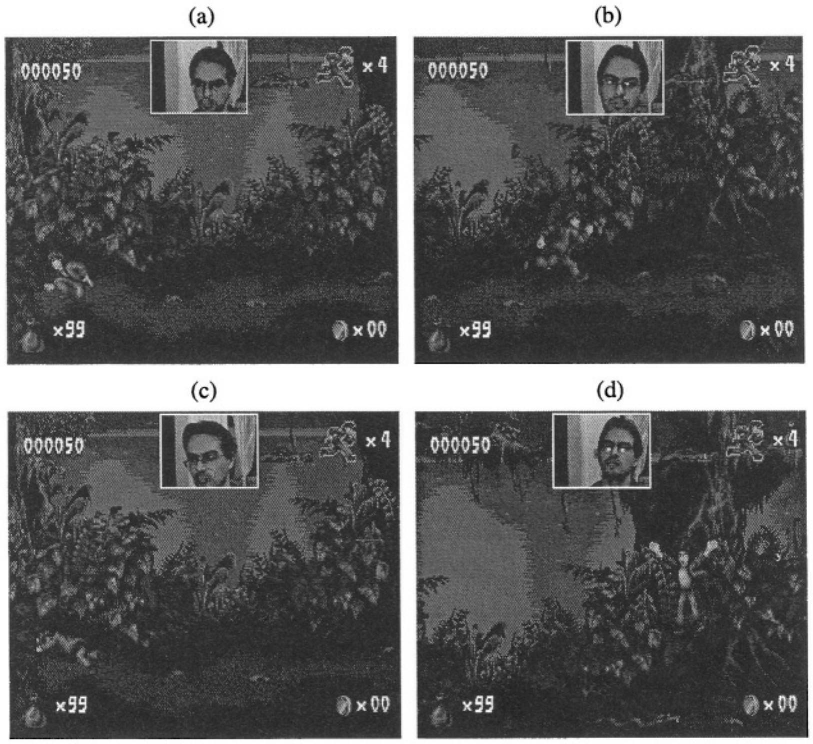


Fig. 3. Experiment with the Pitfall game

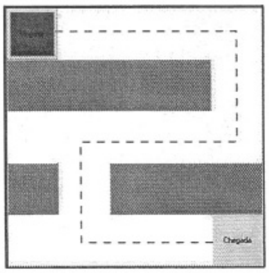


Fig. 4. Maze-like Environment

block (traveller) from the start position to the final position while avoiding walls. We performed 6 trials for which one of 3 users. After each trial, we measured the time spent and the length of the path followed by the traveller.

The walls along the corridor do not block the traveller during its motion. This feature increases the complexity of the task, because it does not constrain the traveller motion. The Figure 5a) shows the evolution of learning for each user during the trials. The axis y corresponds to the path length, in pixels, followed by the traveller and the axis x indicates

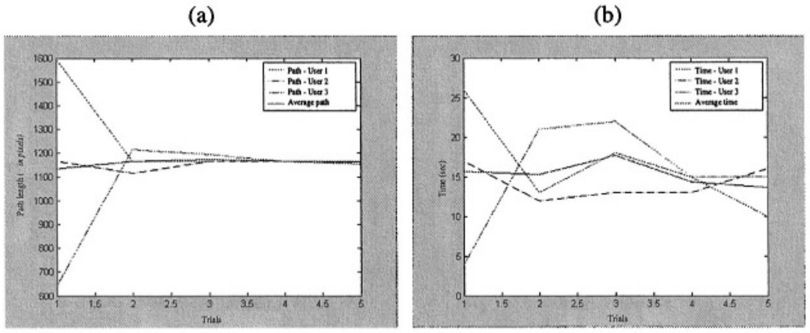


Fig. 5. Experiment results. a) distance as a function of the number of learning trials. b) time spent to perform the task as a function of the number of learning trials

the respective trial. The solid line is the average length of the path produced over trials. The other lines are associated to each user. Observe the path length decreases as the user learns to use the interface. After, only 6 trials, the path length produced by each user is near of the optimal path estimated previously in 1120 pixels.

The average length of the path followed by the traveller after the 6 trials is $\bar{\ell} = 1160.9$ with standard deviation equals to $\sigma_\ell = 169.74$. Figure 5b) illustrates the time spent to perform the task. The users spent an average of $\bar{t} = 15.23s$ to achieve the goal with standard deviation of $\sigma_t = 4.99\ s$.

5 Conclusion

This paper describes an approach based on the Mean Shift Algorithm to control the computer by gestures using a generic webcam. We validated the usefulness of our approach in two different tasks. The first task is specific and consists of controling a game called pitfall. The user controls the treasure hunter in a jungle using the motion of the mouth. The motion of the hunter is limited to 4 specific actions (jump, stoop, move left, move right). We observed that the users quickly learned to control the treasure hunter. This observation was comproved in the second task.

The second task is generic and aims to show the precision and easiness of the interface. In this case, the user was in charge of leading a traveller from a start position to a final position while avoiding walls. The users conducted the traveller to a path close to the optimal path in a short period of time, nearly $15\ s$.

These results are very promising. However, we still have some challenges to overcome. We only use color information to guide the track algorithm. This information is very sensitive to luminosity differences, which can easily generate misclassification. We intend to incorporate texture or spatial information to the color distribution. Furthermore, the track algorithm does not handle faster targets conveniently. It fails if the displacement of the target between two successive frames is bigger than the radius of the search region.

However, it is very important to stress that neither mark nor specific dress was needed in order to make the interface work. This approach is a viable and cheap candidate to help disabled people in daily tasks.

References

1. François Bérard, "The perceptual window: Head motion as a new input stream," in *IFIP Conference on Human-Computer Interaction,* 1999, pp. 238–244.
2. Sébastian Grange, Emilio Casanova, Terrence Fong, and Charles Baur, "Vision-based sensor fusion for human-computer interaction," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS,* 2002.
3. Kentaro Toyama, ""look,ma - no hands!" hands-free cursor control with real-time 3d face tracking," in *Workshop on Perceptual User Interfaces,* 1998.
4. Nan Li, Shawn Dettmer, and Mubarak Shah, "Lipreading using eigensequences," in -, , Ed., -, vol. - of -, pp. –.
5. James W. Davis and Serge Vaks, "A perceptual user interface for recognizing head gesture acknowledgements," -, 2001.
6. Atsushi Nishikawa, Toshinori Hosoi, Kengo Koara, Daiji Negoro, Ayae Hikita, Shuichi Asano, Haruhiko Kakutani, Fumio Miyazaki, , Mitsugu Sekimoto, Masayoshi Yasui, Yasuhiro Miyake, Shuji Takiguchi, and Morito Monden, "Face mouse: A novel human-machine interface for controlling the position of a laparoscope," *IEEE Transactions on Robotics and Automation,* vol. 19, no. 5, pp. 825–841, 2003.
7. Dorin Comaniciu and Visvanathan Ramesh, "Mean shift and optimal prediction for efficient object tracking," 2000.
8. Comaniciu, Visvanathan Ramesh, and Meer, "Real-time tracking of non-rigid objects using mean shift," in *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* 2000, vol. 2, pp. 142–149.
9. Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer, "Kernel-based object tracking," *IEEE Transactions Pattern Analysis Machine Intelligence,* vol. 25, no. 5, pp. 564–575, 2003.
10. J. Lin, "Divergence measures based on the shannon entropy," *IEEE Transactions on Information Theory,* vol. 37, pp. 145–151, 1991.
11. Dorin Comaniciu and Peter Meer, "Mean shift analysis and applications," in *ICCV(2),* 1999, pp. 1197–1203.

Projected Fringe Technique in 3D Surface Acquisition

Carlos Díaz and Leopoldo Altamirano

National Institute of Astrophysics, Optics and Electronics, Luis Enrique Erro # 1, Santa Maria Tonantzintla, Puebla, 72840, Mexico
{cdiaz, robles}@inaoep.mx

Abstract. There have been increasing interest in 3D surface contouring technologies over the last decade. 3D surface contouring techniques have numerous applications in design and manufacturing, 3D contouring technology is also used in reverse engineering where construction of a CAD model from a physical part is required. We propose a system based in the projected fringe technique. We present here early results of a potentially fast, efficient and reliable method to extract the surface contour of 3D external objects. A View Sonic PJ551 digital projector is used as active light source, and a DFK 50H13 CCD camera with a DFG/LC1 grabber are used to grab images. One advantage of projected fringe technique is that allows acquiring dense 3D data in few seconds.

1 Introduction

Conventionally, coordinate measurement machines (CMM's) have been used for measure 3D surface coordinates. CMM's are well-known and widely accepted in industry, but the major drawbacks of these methods is that they first, requires point-by-point scanning, which is time consuming and difficult to reach the requirement of on-line measurement; moreover, requires mechanical contact during measurement, which could produce wear of the probe itself and damage of the measured surface. This hinders their utilization to delicate surfaces, such as optical surfaces, thin films, and silicon wafers.

Optical 3D sensors measure the shape of objects without the need to physically probe surfaces. They are faster, cheaper and provide a higher measurement density than traditional techniques. Various different optical shape acquisition methods have been developed for a wide range of applications, including inspection, robotic vision, machine vision, medical diagnostics, large infrastructure inspection (buildings, bridges, roads, and tunnels) and corrosion inspection.

One of these methods is projected fringe technique. Particularly the projected fringe technique provides dense and reliable surface acquisition by projecting fringe patterns and recover phase information by phase shifting interferometry (PSI).

2 State of the Art

Projected fringe technique is an extension of triangulation [12] for out-of-plane and topography measurement. The fringe patterns are projected on the object surface and these are distorted in accordance with the object height. Unlike the projection moiré techniques, the distorted fringe patterns are directly captured by a CCD camera and the surface height can be reconstructed from the deformed fringes, instead of using a reference grating to create fringes [11, 14]. The main advantage of this technique is that it is very easy to setup and does not require intensive calculation.

2.1 Bases of the Projected Fringe Technique

Digital fringe projection is based in traditional PSI techniques, but fringe projection has advantages in the phase-shifting accuracy, system simplicity as well as measurement speed. PSI is not a specific optical hardware configuration but rather a data collection and analysis method that can be applied to a great variety of testing situation.

An active light source projects fringe patterns on the test object, the varying depths of the objects surface cause phase variations on the pattern projected. These phase changes are used to find out the surface coordinates of the objects to be measurement using PSI and phase unwrapping techniques, [15,9,16].

The basic equation for two-beam interference is

$$I(x,y) = I_a + I_b \cos(\theta(x,y)) \,. \tag{1}$$

or

$$I(x,y) = I_{avg}(1 + \gamma \cos(\theta(x,y))) \,. \tag{2}$$

Let $\theta(x,y)$ be equal to a constant value, δ, and a variable term, $\phi(x,y)$ that depends upon position x, y. The intensity can be written as:

$$I(x,y,\delta) = I_{avg}(1 + \gamma \cos(\phi(x,y) + \delta)) \,. \tag{3}$$

While the equation 3 is the most common way of writing the intensity distribution for two-beam interference, it is often convenient to rewrite $\cos(\phi + \delta)$ as a product of $\cos(\phi)\cos(\delta)$ and $\sin(\phi)\sin(\delta)$. That is:

$$I(x,y,\delta) = I_{avg} + I_{avg}\gamma \cos(\phi(x,y))\cos(\delta) - I_{avg}\gamma \sin(\phi(x,y))\sin(\delta) \,. \tag{4}$$

Letting $a_0 = I_{avg}$, $a_1 = I_{avg}\gamma\cos(\phi(x,y))$, and $a_2 = -I_{avg}\gamma\sin(\phi(x,y))$ we can write:

$$I = a_0 + a_1 \cos(\delta) + a_2 \sin(\delta) \,. \tag{5}$$

It is important to note that

$$\tan(\phi) = \frac{-a_2}{a_1} \,. \tag{6}$$

and

$$\gamma = \frac{\sqrt{a_1^2 + a_2^2}}{a_0} \,. \tag{7}$$

Equations 5 and 6 are the two most useful in PSI.

Many algorithms have been developed for PSI, such as 3-step algorithm, least-squares algorithms, Carré algorithm and Hariharan algorithm. The major difference between these algorithms is how the reference phase is varied and the numbers of times and the rate at which the interference pattern is measured. More detailed descriptions about these algorithms are given in [13].

2.2 Phase Unwrapping Algorithms

In most fringe analysis methods, the arctangent function is used to obtain the phase of the fringe pattern. This function returns values that are defined between the limits π and $-\pi$. Hence, the result is given modulo 2π and discontinuities, of value near to 2π, appear in the resulting phase distribution [6]. The 2π discontinuities should be removed, and the process of removing these discontinuities is called unwrapping [8]. Many different algorithms exist, but a correct solution is not guaranteed, and very long execution times are often involved [5].

Phase unwrapping process is simple, however, things can be very complicated because of all kinds of error sources, especially when an automated phase unwrapping process is required. The error sources that arise most frequently in a fringe pattern are as follows:

1. Background or electronic noise produced during data acquisition.
2. Low data modulation points due to low surface reflectivity.
3. Abrupt phase changes due to surface discontinuities or shadows.
4. Violation of the sampling theorem.

Most phase unwrapping algorithms can handle (1) and (2). (4) can be avoided by changing system setup. For error source (3), one needs to have a priori knowledge of the object or to use special techniques. Otherwise (3) will result in path-dependent phase unwrapping which is unacceptable.

A major goal of fringe analysis is to automate the phase unwrapping process. Automatic techniques are essential if systems are to be run unsupervised or high speed processing is in demand. The following sections briefly review some of the representative phase unwrapping techniques.

1. *Phase Fringe Scanning Method.* Greivenkamp proposed this method in [7]. A horizontal line of the phase image is unwrapped first. Then starting at each point on this line, the whole phase map is unwrapped vertically. This is the most straightforward method of phase unwrapping and therefore is the fastest method among all phase unwrapping techniques. Unfortunately, this method can not handle phase images of objects with areas of low surface reflectivity.
2. *Phase Unwrapping by Sections.* Arevallilo and Burton developed this algorithm [1,3]. An image can be subdivided into four sections. If a section is larger than 2×2 pixels, this section is further subdivided into four sections. It is easy to unwrap a 2×2 area and two areas can be connected by checking their common edge. After subareas have been unwrapped, they are joined together. Points on the edge are traced to judge if a shift should be made, up

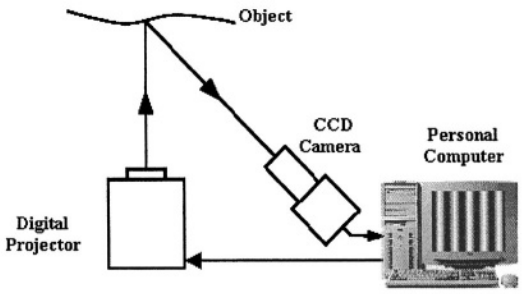


Fig. 1. Layout of the measurement system

by 2π, down by 2π, or no shift according to certain weighting criterion. This method tends to provide global unwrapping optima but has the complexity to deal with error source areas. The weighting criterion for connecting sections is also hard to be used as a general criterion.

3. *Phase Unwrapping by Using Gradient.* This method was proposed by Huntley [10] in an attempt to solve the problem of surface discontinuity. Since a large phase difference between any two adjacent pixels increments the possibility of a discontinuous phase change, this algorithm always tries to choose the direction of the smallest gradient to unwrap phase. The phase at each pixel is compared with its 8 neighboring pixels. The direction in which the phase difference is the smallest is taken to unwrap the phase.

   The drawback of this algorithm is that invalid pixels are phase unwrapped eventually, which introduce phase unwrapping errors. One solution to this problem is to set a threshold phase difference to reject invalid pixels due to noise or phase discontinuity. The threshold has to be flexible in order to adapt to different circumstances, which makes automatic phase unwrapping difficult.

   Using the first phase difference may result in misjudgment in choosing the right unwrapping direction. In [2,4] is proposed the second order difference method which is used to improve the performance. Phase unwrapping based on least gradient does provide the most logic way for phase unwrapping. However the method may not be able to unwrap all the pixels of interest automatically and to handle zones of high curvature of phase map.

3 The Shape Measurement Method

Our method works in the following manner: 1. a digital projector generates the active light source; 2. phase changes due to variant surface depths will cause phase changes in the projected active light on the test object; 3. a CCD camera and an image grabber card are used to capture images. A sketch of the optical system is showed in the figure 1.

3.1 Our Phase Shifting Algorithm

In our research, we use a 7-step algorithm, due to reduces errors likes incorrect phase shift between data frames, vibrations, detector non-linearity, quantization errors, etc. The drawback of this algorithm is that requires 7 data frames. The 7-step algorithm is described as follow:

$$\begin{aligned} t &= 1,2,3,4,5,6,7; \\ \delta(t) &= (t-3)(\pi/2), \\ I(i,j,t) &= I(i,j)_{avg}(1+\gamma\cos[\phi(i,j)+\delta(t)]. \end{aligned}$$

By solving the above 7 equations, we can obtain the unknown phase at each point:

$$\phi = \arctan\frac{4(I_2 - 2I_4 + I_6)}{-I_1 + 7I_3 - 7I_5 + I_7}\,. \tag{8}$$

and the data modulation:

$$\gamma = \frac{2\sqrt{16(I_2 - 2I_4 + I_6)^2 + (I_1 - 7I_3 + 7I_5 - I_7)^2}}{I_1 + 4I_2 + 7I_3 + 8I_4 + 7I_5 + 4I_6 + I_7}\,. \tag{9}$$

3.2 Proposed Phase Unwrapping Algorithm

The phase unwrapping by using gradients approach provides the most logic way for phase unwrapping. The algorithm proposed in this research takes advantage of this approach and uses additional information obtained in data modulation to eliminate errors in the phase image. First of all, part of the fringe images may be saturated due to specular reflection. These pixels have to be excluded from the phase unwrapping process because they do not provide correct phase information. Also, the object may not fill the entire image and it may also have holes on its surface. This often results in a phase image with substantial background noise. All the pixels representing the background should also be excluded from the phase unwrapping process. These saturated pixels and background pixels are identified by calculating the data modulation at each pixel. Assigning a threshold value to the data modulation, a generated mask defines the areas of valid and invalid pixels. Phase unwrapping is done only at valid pixels.

The whole procedure can be summarized as the following steps:

1. *Generate Mask.* We generated a mask image M based on the data modulation calculation. This mask image indicates whether a pixel is valid or not. Only valid pixels are processed in the automatic unwrapping procedure. Figure 2, the black area represents an area of invalid pixels of the phase wrap distribution in figure 6(a).
2. *Calculate Reliability Values.* In this step, we based upon the gradients. Those points with the lowest module 2π gradients with respect to their neighbors are determined to be the best points, therefore, these points are processed first.

Fig. 2. Binary mask, black area represents invalid pixels

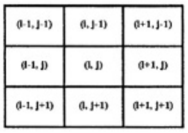


Fig. 3. Calculation of the second derivative in an image

Second derivative would provide a measurement for the degree of concavity/convexity of the phase function. By use of second derivative a better detection of possible inconsistencies in the phase map is provided.

The calculation of second differences for pixels in an image can be explained with the aid of figure 3. To calculate the second difference for a pixel in an image, the values of its orthogonal and diagonal neighbor's in a 3 × 3 window are required. The second difference D of an (i,j) pixel can be calculated by the equation:

$$D(i,j) = \sqrt{H^2(i,j) + V^2(i,j) + D_1^2(i,j) + D_2^2(i,j)}\,. \tag{10}$$

where:

$$\begin{aligned}
H(i,j) &= \gamma[\phi(i-1,j) - \phi(i,j)] - \gamma[\phi(i,j) - \phi(i+1,j)]\,,\\
V(i,j) &= \gamma[\phi(i,j-1) - \phi(i,j)] - \gamma[\phi(i,j) - \phi(i,j+1)]\,,\\
D_1(i,j) &= \gamma[\phi(i-1,j-1) - \phi(i,j)] - \gamma[\phi(i,j) - \phi(i+1,j+1)]\,, \textit{ and}\\
D_2(i,j) &= \gamma[\phi(i-1,j+1) - \phi(i,j)] - \gamma[\phi(i,j) - \phi(i+1,j-1)]\,.
\end{aligned}$$

where $\gamma[(.)]$ is a simple unwrapping operation to remove any 2π steps between two consecutive pixels. The second derivative should be calculated for all pixels in M. The reliability R of a pixel is defined as:

$$R(i,j) = \frac{1}{D(i,j)}\,. \tag{11}$$

3. *Edge Computation.* For this research an edge is considered an intersection of two pixels that are connected horizontally or vertically. We only compute edges of pixels in M. An unwrapping path cannot be defined relative to the reliability of the pixels. Instead, it is defined by looking at the value of the reliability of the edges. We define edge's reliability as the summation of the reliabilities of the two pixels that the edge connects:

$$R_{e_{k,l}} = R_k + R_l\,. \tag{12}$$

where $R_{e_{k,l}}$ is the edge's reliability defined by k and l pixels.

4. *Initialize Groups.* Initially all valid pixels are considered not belonging to any group. Not valid pixels are considered belonging to group 0, that it won't be considered in automatic phase wrap process.
5. *Unwrapping Path.* In unwrapping path those valid edges with higher reliability are unwrapped first. The edges are stored in an array and sorted by value of reliability. When the process is performed pixels forms group of pixels. When is analyzed a edge in the unwrapping process, three cases are possible:
   (a) Both pixels have not been unwrapped before. The pixels are unwrapped with respect to each other and gathered into a single group of unwrapped pixels.
   (b) One of the pixels has been processed before, P_1 but the other has not, P_2. P_2 is unwrapped with respect to P_1 and added to the P_1's group.
   (c) Both pixels have been processed before. If they do not belong to the same group, the two groups need to be unwrapped with respect to each other. The smallest group is unwrapped with respect to the largest group.

If the wrapped phase map is composed of disconnected valid parts, the above steps finish all parts without depending on where the starting point is located. Certain relationships among these separate parts have to be predetermined in order to connect the whole surface.

4 Measurements Results

As mentioned previously, we use CCD camera to capture seven images with fringes, which shift $\frac{\pi}{2}$ between each other, then we do phase wrapping and unwrapping to get the 3D phase map, finally we display the 3D representation of the phase unwrap distribution. Figure 4(b) to (h) shows the fringe patterns, figure 4(i) is the wrapped phase map, figure 4(j) is 3D phase map and figure 4(a) is the 2D image. This measurement results shows that our measurement system is capable of measuring static 3D objects, and get satisfactory results.

We measured a plate with small variations onto the surface. Since this system has not been calibrated yet, the unwrapped phase map represents an unsealed approximation of the real shape, i.e., the measurement result was relative phase values, not absolute coordinates. From the unwrapped phase map, it can be seen that proper measurement results were obtained. The execution time of the proposed system varies from image to image and depends on the particular phase distribution being analyzed. The tested images are 640×480 pixels in size. The proposed system has been executed on a PC system. The PC contains a Pentium 4 processor that run at 2 GHz clock speed. The memory on this PC is 256 Mbyte DDR. The execution time is in the order of two second on average. The execution time to wrap and unwrap the image shown in figure 4 is 2.234 seconds. Processing average time can see in table 1.

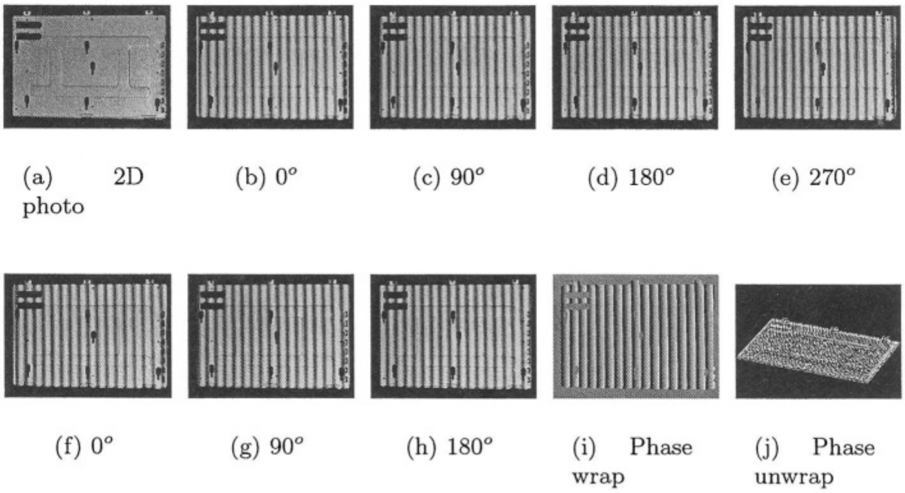


Fig. 4. 7-step phase shifting measurement example 1

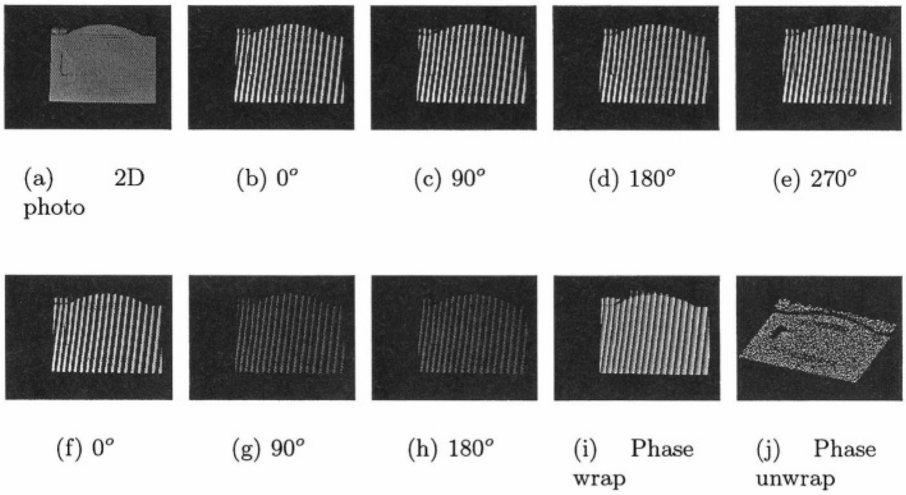
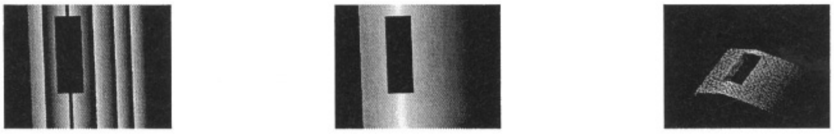


Fig. 5. 7-step phase shifting measurement example 2

Table 1. Processing average time to recover phase information

Process	Average time
Data acquisition	0.233s
Phase wrap	0.235s
Phase unwrap	1.984s
Total time	2.452s



(a) Wrapped phase distribution.

(b) Unwrapped phase distribution.

(c) 3D representation of the resulting phase.

Fig. 6. 7-step phase shifting measurement example 3

5 Conclusions

A novel high-speed phase shifting technique with a potential measurement speed is presented. In this research, we develop methods to implement a high precision 3D measuring systems. The advantages of the proposed approach include:

1. The measuring speed is fast, which is suitable for on-line spot measuring;
2. Every picture element can be considered as a sample point thus achieving a high image resolution;
3. It is independent to the object surface color because it uses grayscale patterns; and
4. This method does not require cumbersome optical setup.

Seven phase shifted fringe patterns can be obtained within 0.23s, and execution time to get phase model is in the order of two seconds on average, so we can use this approach in on-line inspection. Some experiments were done to test the performance of this technique and satisfactory results were obtained.

References

1. M. Arevallilo Herraéz, D. R. Burton, and D. B. Clegg, "Robust, simple, and fast algorithm for phase unwrapping", Applied Optics, 1996, 35, 5847-5852.
2. M. Arevallilo Herraéz, D. R. Burton, M. J. Lalor, and M. A. Gdeisat, "Fast two-dimensional automatic phase unwrapping algorithm based on sorting by reliability following a noncontinuous path", Applied Optics, 2002, 41(35), 7437-7444.
3. M. Arevallilo Herraéz, M. A. Gdeisat, D. R. Burton, and M. J. Lalor, "Robust, fast, and effective two-dimensional automatic phase unwrapping algorithm based on image decomposition", Applied Optics, 2002, 41 (35), 7445-7455.
4. R. Cusack, J. M. Huntley, and H. T. Goldrein, "Improved noise-immune phase-unwrapping algoritm", in Applied Optics, 34 (5), 781-789, 1995.
5. D. C. Ghiglia, and M. D. Pritt, "Two-Dimensional Phase Unwrapping: Theory, Algorithms, and Software", Wiley, 1998, New York.
6. C. Gorecki, "Interferogram analysis using a Fourier transform method for automatic 3D surface measurement", Pure and Applied Optics, 1(2), 1992, 103-110.

7. J. E. Greivenkamp and J. H. Bruning, “Phase Shifting Interferometers”, in Optical Shop Testing, D. Malacara, 2nd Ed. 501-598, Wiley, New York, 1992.
8. P. Hariharan, “Interferogram Analysis: Digital Fringe Pattern Measurement Techniques”, in Applications of interferogram analysis, by W. R. Robinson and G. T. Reid, Bristol, Institute of Physics Publishing, ISBN 075030197X, 1993, pp 262-284.
9. Q. Hu, “3-D Shape Measurement Based on Digital Fringe Projection and Phase-Shifting Techniques”, Ph. D. Thesis. State University of New York. 2001.
10. J. M. Huntley and H. O. Saldner, “Error reduction methods for shape measurement by temporal phase unwrapping”, in Journal of Optical Society of America, 14 (2), 3188-3196, 1997.
11. G. Indebetouw, “Profile measurement using projection of running fringes”, Appl. Opt., 17(18), 1978, 2930-2933.
12. Z. Ji and M. C. Leu, “Design of optical triangulation devices”, SPIE: Optics and Laser Technology, 1989, 21(5), 335-338.
13. D. Malacara, “Optical Shop Testing”, Wiley, New York, 2nd edition, ISBN 0-471-52232-5.
14. M. M. Shaw, D. M. Harvey, C. A. Hobson and M. J. Lalor, “Non-contact ranging using dynamic fringe projection”, Proc. SPIE, 1163, 1989, 22-29.
15. J. C. Wyant, “Phase-Shifting Interferometry”, Technical Report. Optical Sciences Center, University of Arizona, 1998.
16. S. Zhang, “High-speed 3-D Shape Measurement Based on Digital Fringe Projection Technique”, State University of New York, 2003.

Optimized Object Recognition Based on Neural Networks Via Non-uniform Sampling of Appearance-Based Models

Luis Carlos Altamirano and Matías Alvarado

PIMAyC, Instituto Mexicano del Petróleo,
Eje Central Lázaro Cárdenas 152,
San Bartolo Atepehuacán, 07730, México DF
(laltamir, matiasa)@imp.mx

Abstract. The non-uniform sampling of appearance–based models supported by neural networks is proposed. By using the strictly required images –obtained by applying non-uniform sampling- for modeling an object, a significant time reduction for the training process of neural networks is achieved. In addition, high levels of recognition are obtained.

1 Introduction

For object recognition, the appearance–based modeling has achieved high ratings of recognition, even for objects with complex appearance [1], [2]. The non-uniform sampling of images for building appearance-based models reduces the quantity of required images, and in this way, it allows an important saving of computing time and storage space through the *eigenspaces* technique. Moreover, from the workspace the non-uniform sampling, it also improves the quality of the models [3]. In this paper the synergy of non-uniform sampling based on neural network (NN) is addressed.

The traditional appearance-based approach builds a model of the object through a set of training images taken by uniform sampling from the workspace. However, uniform (non-discriminating) sampling usually requires a large set of training images for object recognition, and then requires a huge quantity of computing time and storage space in order to build the model [4], [1]. Recently, the non-uniform sampling [4] has been proposed as a way to reduce he quantity of the required images, that in turn it implies storage and time saving. The integration of non-uniform sampling and the *eigenspaces* technique [5] has been successfully proved. Herein, the use of non-uniform sampling on neural networks is analyzed.

Several works that use NN to recognize objects from their appearance have been proposed. For example, in [6] neural networks were used for face recognition; the wavelet approach was employed in [7] for obtaining models from de objects for training neural networks; in [8] regularization networks trained with few views of the objects were used. However, it is unknown which are the appropriate images that the neural networks should be trained. The experimental results obtained here, point out that is possible to reach an important reduction in the computing time, besides the high levels of recognition, employing non-uniform sampling inside neural network paradigm.

The rest of the paper is organized as follows: in Section 2, preliminaries concepts are exposed. In Section 3, the synergy of non-uniform sampling and neural networks is proposed. In Section 4, experimental results about the proposal done in previous section are introduced. Finally, conclusions and future work are discussed.

2 Preliminaries

To create appearance-based models, it starts with the images acquisition of the objects to be modeled. The images are taken around the object over a plane. The object rotate in front of the camera through an angle θ, respect to an initial position to acquire the i-th image I_i, with $\theta_{i-1} < \theta_i$. These images define the trajectory known as the appearance of the object. For the reason of simplicity most of the previous works take $\theta_i - \theta_{i-1} = k$, with k a constant [4]. This is known as a uniform sampling. Typical values founded in the reading of k oscillate between 5° and 12°. However, by supposing that k must be a constant is not a well-founded supposition [3], due to a fixed quantity of images to model any object, does not consider the specific characteristics of the object.

2.1 Non-uniform Sampling

The main idea for non-uniform sampling is that the required number of images in order to model the object appearance is different for each object zone (parts of the model). Usually, there are zones in the appearance of the object that can be represented using fewer images than other ones. This consideration makes to achieve a substantial reduction of computing process and time storage with respect to uniform sampling. Technically speaking, it means that there are zones in the appearance of the object that presents a quasi-linear behavior, such that they can be approximated by a small quantity of images. On the contrary, there are zones where the behavior is not quasi-lineal and they will require a bigger quantity of images in order to rightly represent these zones.

In [3], the objects appearance is approximated by means of piecewise lineal interpolation in an interval of error $[-\varepsilon, \varepsilon]$. Thus, important reductions in computing time and storage space, in addition to high levels of recognition are achieved by combination of non-uniform sampling and the *eigenspaces* technique.

3 Neural Networks

3.1 Neural Networks and Dimensional Reduction

Although training the neural network can directly use the acquired images of the object, it is desirable to do a reduction process of dimensionality to avoid the use of a neural network of very big dimensions. A usual manner to reduce the dimensionality is obtained through the *Hotelling Transformation.* This transformation allows the representation of an image inside the set of training images correlated, through a vector with few (not correlated) components, so-called principal components [9]. The application of Hotelling transformation is computationally intense and it usually depends on the number of training images to be processed. By applying the Hotelling transformation, a projection of the original images with few components are obtained, which can be used to train a neural network. Because this reduction, the time for training the network and storage space is decreased.

3.2 Neural Networks and Non-uniform Sampling

In this work, the use of the image set obtained by non-uniform sampling to train a neural network for objects recognition is proposed. Before, in order to spend less training time, applying the Hotelling transformation reduces the dimensionality of the image set. In this work, the multilayer perceptron is used due to its successful applicability in several pattern recognition tasks [10]. This kind of neural networks allows discriminating between complex sets whenever the number of neurons is enough.

4 Experimental Results

The proposal was implemented in a software system developed in MATLAB. The objects used to test the system efficiency are of different appearance complexity, namely, different textures, colors and shapes. Some of them were taken from the COIL (Columbia Object Image Library) [2] that is a classical reference. Proofs were

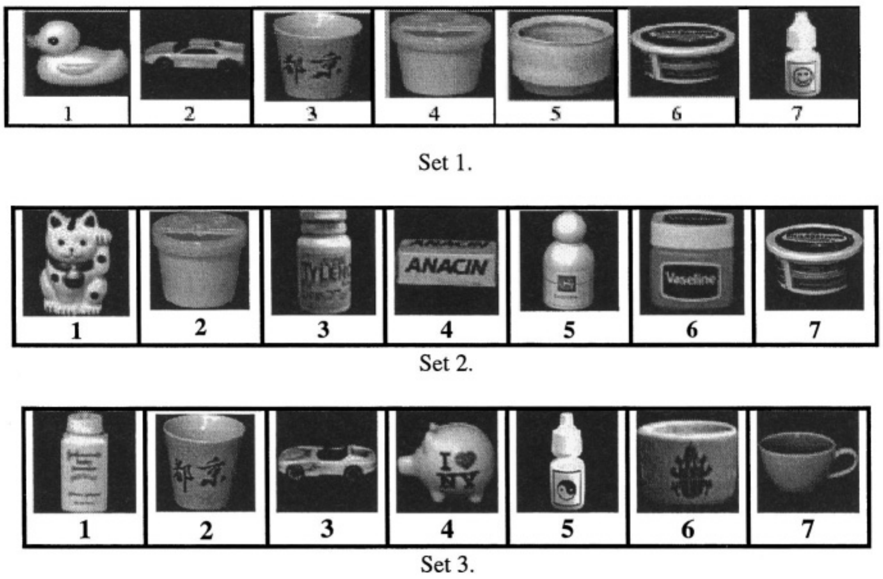


Fig. 1. Modeled/recognized objects

done with more than 50 different objects, but only results for the objects shown in Figure 1 were documented.

The objects were segmented from the original image in order to eliminate the background. The images were normalized in scale to be 128x128 pixels in grey tones (both, uniform and non-uniform sampling). For uniform sampling, the rotation angle between consecutive images was 10° (36 images per object).

For the non-uniform sampling, in order to determine the quantity of necessary images to model the studied objects, the highest precision allowed by the rotation system was applied. The highest precision is such that the system can build the model of the object, without additional images taken with a minor angle to the established rotation limit (10°). The required number of images per object is shown in Table 1. Notice the important reduction respect to a uniform sampling (36 images per object).

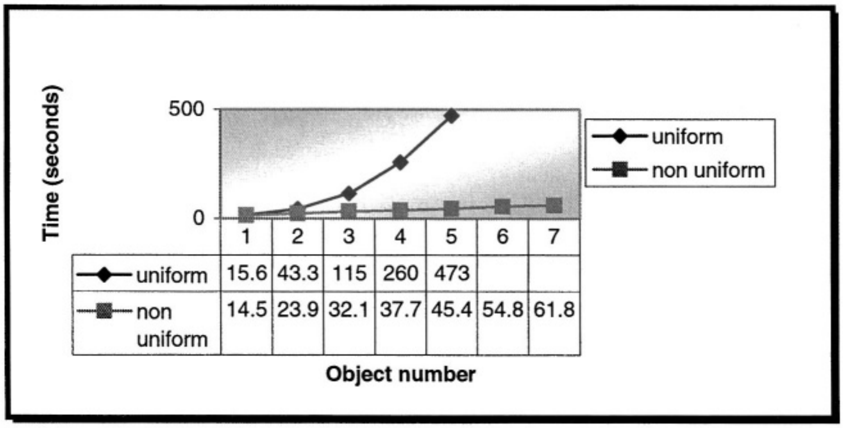
Table 1. Required images to model/recognize objects in Figure 1 using non-uniform sampling

	Object number						
	1	2	3	4	5	6	7
Set 1	32	18	11	6	6	9	4
Set 2	26	6	12	23	13	22	9
Set 3	25	11	18	22	3	18	26

The Hotelling transformation was calculated for each set of objects in Figure 1. The required time to calculate 20 eigenvectors of images with a 1.2 Ghz Celeron-based PC is shown in Figure 2. The usage of non-uniform sampling drastically reduces the processing time. Results for uniform sampling with more of 5 objects are not shown due MATLAB could not process the matrix covariance required (over to 180x180 elements).

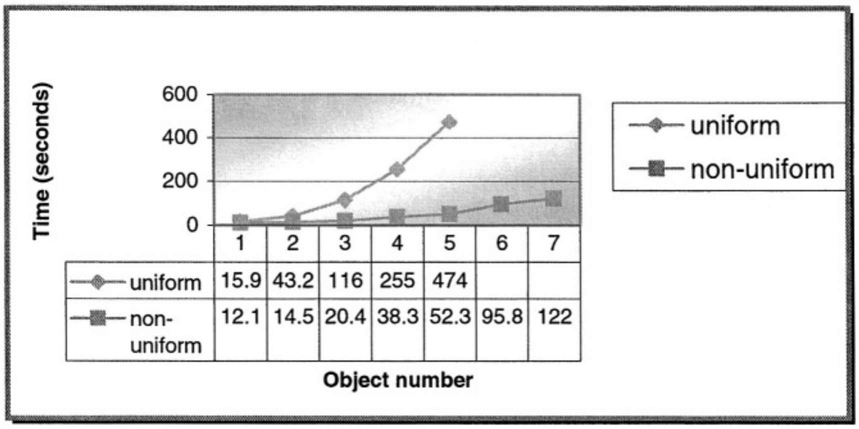
The used NN has 2 hidden layers, with 9 and 3 neurons respectively. For these experiments, the 20 eigenvectors calculated in the previous stage were used. The results of the training stage with the previous values for both, uniform and non-uniform sampling are shown in Table 2. Notice that the training process is more effective using non-uniform sampling.

The NN recognizing level is determined as the last stage. A total of 576 images (72 images x 8 objects) were used in order to the network determines which object class belongs each one. This image set included known (training images) and unknown images of the studied objects. The percentages of recognition obtained for each of the objects with the trained network are shown in Table 3. They are close to 100%. Thus, based on NN, the non-uniform sampling gets reduction of processing time as well as storage space; the recognition levels with uniform sampling are similar, but they are more expensive as commented before.



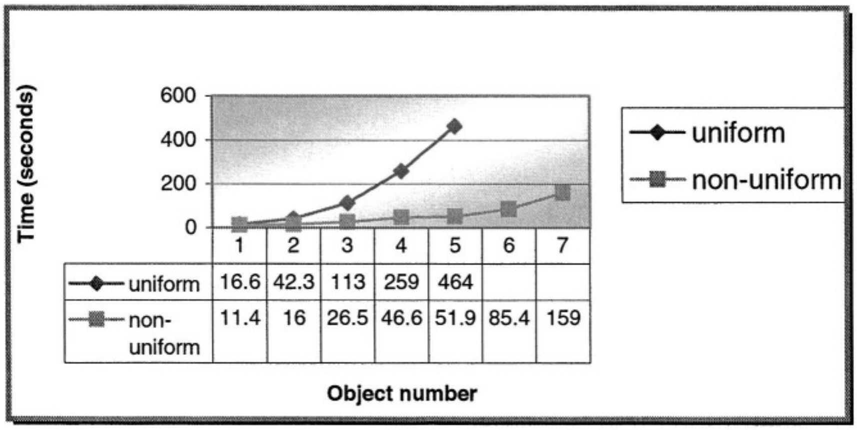
	1	2	3	4	5	6	7
uniform	15.6	43.3	115	260	473		
non uniform	14.5	23.9	32.1	37.7	45.4	54.8	61.8

Set 1.



	1	2	3	4	5	6	7
uniform	15.9	43.2	116	255	474		
non-uniform	12.1	14.5	20.4	38.3	52.3	95.8	122

Set 2.



	1	2	3	4	5	6	7
uniform	16.6	42.3	113	259	464		
non-uniform	11.4	16	26.5	46.6	51.9	85.4	159

Set 3.

Fig. 2. Required time to reduce dimensionality

Table 2. Required time for training the neural network

	Uniform sampling (just 5 objects)	Non-uniform sampling
Set 1	236.95 seconds	25.54 seconds
Set 2	226.46 seconds	29.55 seconds
Set 3	333.29 seconds	33.59 seconds

Table 3. Recognition rating

	Object number						
	1	2	3	4	5	6	7
Set 1	100%	83.33%	100%	100%	100%	100%	100%
Set 2	81.94%	100%	100%	100%	100%	100%	100%
Set 3	100%	100%	83.33%	100%	100%	100%	100%

5 Concluding Remarks

The usage of non-uniform sampling based on NN for object recognition with appearance-based models is presented. The experimental results point out the advantage of non-uniform approach in combination with the neural networks flexibility: it allows a reduction in time of training together with the high index of recognition. For further studies, non-uniform sampling used with NN like learning rules, will probably produce a higher reduction of training time, but keeping high levels of recognition. At the moment, this work is the best effort.

References

1. Nayar, S., Murase, H., Nene, S.: Parametric Appearance Representation. Early Visual Learning, Edited by Shree K. Nayar and Tomaso Poggio, New York Oxford, Oxford University Press. (1996)
2. Nayar, S.K., Nene, S.A., Murase, H.: Real-Time 100 Object Recognition System. Proceedings of ARPA Image Understanding Workshop, San Francisco (1996)
3. Altamirano, L.C., Altamirano, L., Alvarado, M.: Non-uniform Sampling for Improved Appearance-Based Models. Pattern Recognition Letters. Vol. 24, Issue 1-3, Elsevier Science, The Netherlands (2003) 529-543
4. Stevens, M.R., Beveridge, J.R.: Integrating Graphics and Vision for Object Recognition. Kluwer Academic Publishers (2000)
5. Murase, H., Nayar, S.K.: Visual Learning and Recognition of 3-D Objects from Appearance. International Journal of Computer Vision, Vol. 14, No. 1, (1995) 5-24
6. Gong, S., Ong, E., Loft, P.: Appearance-Based Face Recognition under Large Head Rotations in Depth. Proceedings of Computer Vision ACCV'98. Third Asian Conference on Computer Vision. Hong Kong, China, Vol. II (1998) 679-686
7. Pauli, J., Benkwitz, M., Sommer, G.: RBF Networks Appearance-Based Object Detection. Proceedings of ICANN, Paris, Vol. 1, (1995) 359-364

8. Poggio, T., Beymer, D.: Regularization Networks for Visual Learning. Early Visual Learning. New York. Oxford University Press (1996)
9. Oja, E.: Subspace Methods of Pattern Recognition. Res. Studies Press, Hertfordshire, (1983)
10. Hilera, J.R., Martínez, V.J.: Redes Neuronales Artificiales. Fundamentos, Modelos y Aplicaciones (in Spanish). AlfaOmega/Ra-Ma, (2000)

A Statistical Validation of Vessel Segmentation in Medical Images

Francisco L. Valverde[1], Nicolás Guil[2], Enrique Domínguez[1], and Jose Muñoz[1]

[1] Department of Computer Science,
[2]Department of Computer Architecture,
University of Malaga, 29071 Malaga, Spain
valverde@uma.es

Abstract. Validations about contour detection for segmentation in medical images are traditionally carried out by visual human inspection. In this context, automatic and formal validation of results becomes essential. A formal non parametric statistical framework for validation of vessel detection in medical images is presented. To obtain a formal validation of the segmentation process, a statistical test about the vessel contour is developed. The test proposed measures the way that the obtained (segmented) vessel fits in the theoretical vessel model avoiding the human inspection used in other methods. This test is used in this paper to validate a segmentation method for vessel detection in medical images that cope with high noisy images for mammograms. Results in this paper show that our segmentation model is validated for vessel detection with a signification value less than 0.05.

1 Introduction

Segmentation process in image analysis subdivides an image into its constituent parts or objects. One important fact in the development of segmentation techniques is that no general theory exists [1;2], and *ad hoc* solutions are traditionally proposed [3]. Although some initial attempts in the direction of a unified theory were reported by [4], this problem is far from being solved, and none of the developed techniques is generally applicable. Given a particular application, finding the appropriate segmentation algorithm is still a problem [1]. Performance evaluation and comparison of competing techniques become indispensable in this context. We will present a general framework for segmentation evaluation and validation of vessels in medical images.

This paper is organized as follows: In the next section, a brief review of previous work on segmentation evaluation is given. The limitations of these proposed methods and some open problems are pointed out. In section 3 our geometrical vessel model and segmentation method based on snake for vessel detection in medical images is described. Our validation method based on a statistical test is presented in Section 4, where a formal statistical model of vessel is proposed. In section 5, results of evaluation and validation are presented. Finally, in Section 5, the advantages of our approach are summarized.

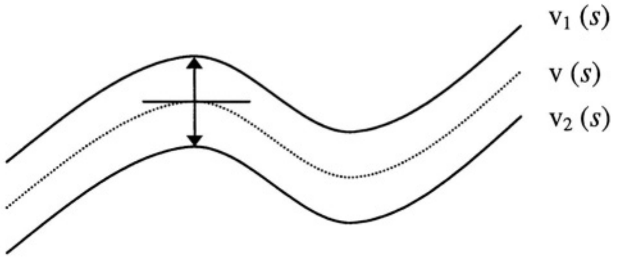


Fig. 1. Vessel geometric model. Two parallel lines across the vessel

2 Evaluation Methods for Segmentation Techniques

In contrast to the abundant number of existing segmentation techniques, only a few methods for evaluation and comparison have been proposed. One important reason for this is the lack of appropriate measures for judging the quality of segmentation results and the performance of the algorithms applied. Zhang [5] classified evaluation methods in three categories: *analytic, empirical* and *empirical with true image.* Early proposed methods for *analytic evaluation* were developed by [6] and [7], where the algorithm is treated directly and no output image is considered. [4], [8], [9], [10], [11], and [12] presented alternative methods based on *empirical evaluation* of the output image without a true image. In addition, [13], [14], [15], [16], [17], [3], [18], and [19] proposed several methods based on the comparison of object features between true and obtained segmented images, that is, an evaluation of the output image considering the true image as reference *(empirical evaluation with true image).*

These evaluation methods also have some common limitations, and several general problems can be pointed out. First, we should stress that there is no unique or standard criterion for segmentation results, because of the lack of a general segmentation theory. Second, because many methods need human inspection to provide a reference, segmentation quality is defined in terms of their correlation with human performance. However, a “pleasing” picture does not necessarily reflect an accurate segmentation. The third limitation is related to the involvement of noise in the image. Normally, in some applications such as satellite images or medical imaging, noise is always present. In those cases, evaluation techniques with a true image cannot be precise because of difficulty of obtain a accurate true image. Some fragment of edges can be lost and the expert human need to extrapolate information to complete the object, in our case the vessel. In this context, a formal validation test based on a statistical model of the object is critic for evaluation [20].

3 Segmentation Method and Vessel Geometrical Model

Segmentation method used in this research work is based on a geometrical deformable model. Deformable models are a useful tool for image segmentation.

They can be classified, according to the information held by the model [21], in local deformable models and global deformable models. Local deformable models manage information to pixel level taking into account only a close pixel neighborhood. However, global deformable models can use information from any location in the image. Local deformable models are faster to converge and they do not need a priory information as a template of the object to be segmented. The most popular approach to local deformable model is the "snake" by *Kass et al.* [22]. Unfortunately, they are very sensitive to noise. On the other hand, global deformable models have slower convergence than local deformable models and usually they need a template of the object. However, global deformable models are more robust and less sensitive to noise than local deformable models.

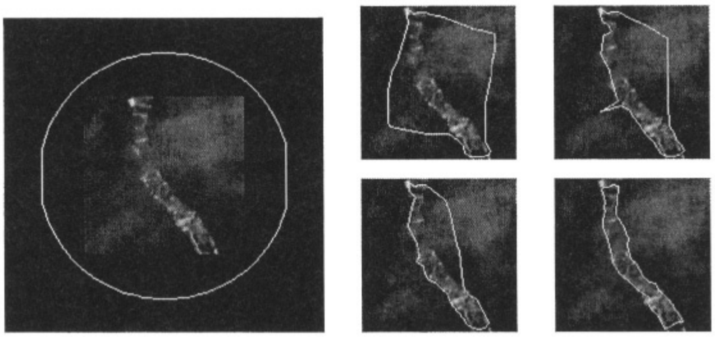


Fig. 2. Initial contour and iterative process (geometric deformable model)

Geometrical deformable models are a particular case of global deformable models that use geometric information of the object. Geometrical models have been used for a number of applications in image segmentation. Continuous geometric models consider an object boundary as a whole and can use the *a priori* knowledge of object shape to constrain the segmentation problem. Our geometric deformable model avoids guided initialization by using a previous local deformable model where no user interaction is required [20].

In the segmentation task, we assume the object to be detected is represented by a contour(x) in an image(z). A polygon (or vector) representation of an object is a representation where the contour is defined by a set of nodes giving coordinates of contour points in a circular (clockwise) manner. Between each node, the contour is defined by a straight line (or some spline-curves).

Geometrically, a 2-D vessel projection can be defined as two parallel edges having a distance between them (width of the vessel) within a range. Let's consider that $v(s)$ with $s \in [a,b]$, is the axis curve of the vessel. Then, the edges of the vessel, $v_1(s)$ and $v_2(s)$ (Figure 1), can be defined as:

$$\begin{aligned} v_1(s) &= v(s) + \lambda/_2 \frac{\nabla v(s)}{\|\nabla v(s)\|} \\ v_2(s) &= v(s) - \lambda/_2 \frac{\nabla v(s)}{\|\nabla v(s)\|} \end{aligned} \tag{1}$$

with $s \in [a,b]$, where $\nabla v(s)$ is the gradient vector of s and λ is the width of the vessel. That is, the two edges and the axis have the same gradient vector in a point s.

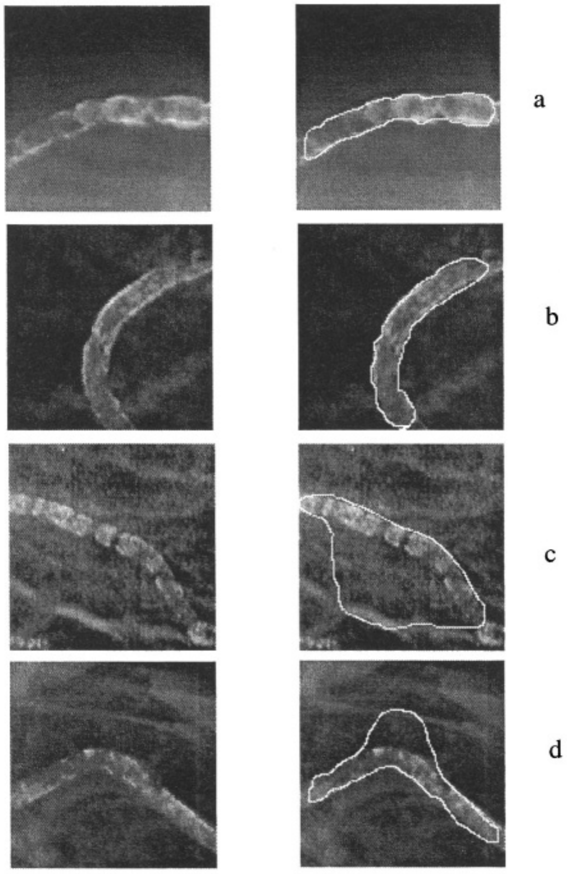


Fig. 3. (Left) original image, (right) vessel segmentation superimposed to the original image (a-b) correct vessel segmentation (c-d) vessel contour is trapped by noise particle

Usually, deformable global models have longer computation time than local deformable models. A critical factor in computation time is the initial contour. Some systems require a human operator to draw an initial contour close to the object. In our model, this requirement is avoided using a initial contour provided by a local deformable model, where no initial contour close to the solution is required. Usually, as initialization of the local deformable model is used a regular shape outside the object. Our method uses a circumference as initial contour (see example of fig 2).

Solutions of the deformable model correspond to minimizing the energy function. Minimization algorithms can be based on dynamic programming [23], variational calculus [22] or iterative exploration [24], which is the most popular approach. We use this approach for our model. This method moves the contour towards the optimal solution. In a iterative process, the initial contour moves towards the solution as shown in Fig 2.

The geometrical deformable contour segmentation has been tested in a database of mammograms by detecting vessels on the images. These images (called Regions of interest ROI) have a size of $L \cdot L$, begin $L =$ 128 pixels with 1024 gray levels. The geometrical deformable model [20;25;26] was applied to a database of 200 ROIs. Some result images are shown in Figure 3. Some resulting contours are shown in figure 3. Note that two of them were trapped by noise.

4 Validation Test

The segmentation method for vessel detection described in section in section 3 was evaluated by a empirical evaluation with a true image [20] (extracted by an expert radiologist for each vessel). However, the true image is only available in testing stage. In this context, an empirical evaluation with no true image for resulting vessel is crucial. For this aim, a. validation test based on a formal statistical model of vessels is developed. This validation test must confirm than the resulting contour corresponds to a vessel with a low significance value.

Our geometric vessel model consist in a two parallel lines as defined in section 3 (see figure 1). However, this model does not correspond exactly with the real vessel in medical images. Small variations on contour points can be found due to noise present in the image and also due to characteristic anatomy of the vessel. Therefore, we develop a statistic model to incorporate these small variations. With this assumption, the vessel model is defined by two almost parallel curves (see figure 4). This characteristic is used to obtain a new statistical vessel model:

Let $v_1 = v(s_1)$, $v_2 = v(s_2), \ldots, v_N = v(s_n)$, $s_1 < s_2 < \ldots < s_n$, n random points from the contour vessel and, $v_1', v_2', \ldots, v_n'$ their frontal respective points. We obtain a set of n values $x_i = \|v_i - v_i'\|$, i=1,2,…,n. Now, we suppose that these values are n independent observations from a symmetric probabilistic distribution, with a mean value of μ and standard deviation σ^2 (very small and not known value).

Now we propose a statistical test to check the null hypothesis.

***H_o*: Distances between frontal points have a symmetric distribution with the same mean and variance values**

The problem of testing of hypothesis may be described as follows: Given the sample point, find a decision rule (function) that will load to a decision to accept or reject the null hypothesis.

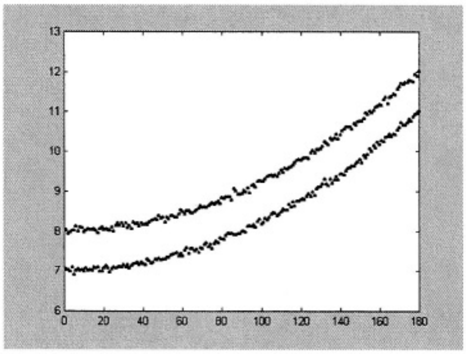


Fig. 4. Statistical vessel model

Let $X_1, X_2,..., X_r, X_{r+1},..., X_{2r},..., X_{(k-1)r},..., X_{kr}$, be kr random variables that denote distance values between consecutive frontal edge pixel from the contour of a vessel. This random variables have a same symmetric distribution with mean μ (unknown) and variance σ^2 (unknown) under the null hypothesis. Next, we define the random variables:

$$Y_i = \frac{1}{r}\sum_{j=1}^{r} X_{(i-1)r+j} - \mu, \quad i = 1,2,\dots,k \tag{2}$$

that have a symmetric distribution with mean 0 and variance σ^2/r.

Now, we form m cluster of k consecutive random variables and define the binary random variable Z_i that takes the values one if all consecutive random variables $Y_{(i-1)h+1}, Y_{(i-1)h+2},..., Y_{ih}$, have taken values greater than 0. That is,

$$Z_i = \begin{cases} 1 & \text{if} \quad \min\{Y_{(i-1)h+1}, Y_{(i-1)h+2},\dots,Y_{ih}\} > 0 \\ 0 & \text{otherwise} \end{cases}, \quad i=1,2,\dots,m \tag{3}$$

where $m = k/h$.

A particular set of values of this random variables is an ordered sequence of m elements (symbols) of two types (0 and 1). A *run* is defined to be a succession of one or more identical symbols which are followed and preceded by a different symbol or no symbol at all. Any particular sample will be a sequence of 0's and 1's which will consist of alternating runs of 0's and 1's. The length of a run is the number of elements in it. Thus, Z_i is the random variable that denotes the symbol in the i-th position. Under the null hypothesis we have $P(Z_i = 1) = 1/2^h$. The null hypothesis would be rejected if there are large runs of the symbol 1. The test proposed is based on the random variable T_m that denotes the number of runs of the symbol 1 of length equal to 2 or greater in an ordered sequence of m elements. Now, we must find the distribution of this random variable under the null hypothesis. We denote $p(m)=P(T_m \leq 1)$. This probability is given by the following recursive relation:

$$p(m) = p(m-1)(1-\frac{1}{2^h}) + p(m-2)(1-\frac{1}{2^h})\frac{1}{2^h} \quad (4)$$

where $p(0)=1$ and $p(1)=1$.

The solution for above recurrent equation is given by expression

$$p(m) = A\alpha^m + B\beta^m \quad (5)$$

where α and β are the solutions of the equation following:

$$x^2 - (1-\frac{1}{2^h})x - (1-\frac{1}{2^h})\frac{1}{2^h} = 0 \quad (6)$$

Coefficients A and B are determined according to initial conditions. When the edge vessel image (ROI) consist of 160 pixels we could take r=4 and h=4, so m=10. In this case,

$$p(m) = p(m-1)\frac{15}{16} + p(m-2)\frac{15}{16}\frac{1}{16} \quad (7)$$

and the solutions of the equation

$$x^2 - \frac{15}{16}x - \frac{15}{16^2} = 0 \quad (8)$$

are α= 0.9963107194 and β=-0.05881071938 and the parameter A= 1.003496546 and B=1-A. So, $p(10)$ = 0.9671.

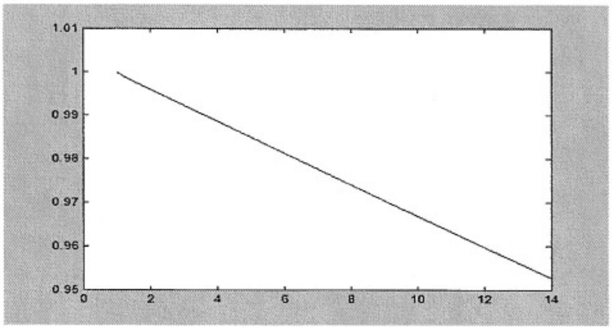


Fig. 5. Graphic representation of p(m)

When the active contour is trapped by noise pixels then a wider area appears in the vessel (see Fig. 3) and this event causes a run of length two or greater of 1's. The statistical test is based on the following decision rule: The hypothesis H_0 is rejected if T_m is greater than one, that is, when appears a run of 1's of length greater than one.

The level of significance is the probability of error of type I, that is, the hypothesis H_0 is rejected when in fact it is true. Thus, this probability is given by 1–$p(m)$. Note that $1-p(10) = 0.0329$. The procedure used in practice is to limit the level of significance to some preassigned between 0.01 and 0.05.

Finally, the mean μ is usually unknown but it can be estimated by the sample mean and the sequence of values $Z_i\, Z_{i+1}$, $i=1,2\ldots, m-1$.

A correct vessel is characterized by small fluctuations of x'_i around the mean vessel with, However, when the result of segmentation is not correct due to particle noise, where the contour is trapped, then the vessel with this area is significantly larger than the mean vessel with (see figure 3.d).

5 Results

A total of 200 ROIs images containing a vessel were used in this study. The snake described in section 3 was used to obtain a vessel contour for every ROI. Every contour was analyzed by a human specialist and a total of 173 ROIs were classified as correct vessel detection and 27 contours were classified as not correct vessel segmentation.

To apply the test to all contours, a sequence of 160 pixels (10 blocks run points, r=4) was chosen. This way the signification level for all contours is the same, 0.0329.

Validation test was positive for all contours classified as correct by the human expert. In addition it was negative for all the contour classified as not correct. That is to say, the 200 ROIs were correctly classified by the test.

The algorithms were implemented in a Pentium III 1.7 Ghz, running Red Hat 7.0 linux operating system.

6 Conclusions

Validation test in medical image segmentation has been usually carried out by human inspection of an expert radiologist. Here, a new method for automatic and formal validation has been presented for vessel segmentation in mammograms. This method is based on a non parametric statistical test that validates a segmented vessel according to a geometrical model of mammogram vessel. A test of run points blocks based on a statistical model of vessel is applied to validate the contour obtained in the segmentation process. Results presented show that all images (100%) were validated correctly by the test.

References

1. Pal, NR, Pal, SK. A review on Image segmentation techniques. Pattern Recognition 29:1277-1294.
2. Zhang, YJ, Gerbrands, JJ. Objective and quantitative segmentation evaluation and comparison. Signal Processing 39:43-54.
3. Yasnoff, WA, Mui, JK, Bacus, JW . Error measures for scene segmentation. Pattern Recognition 9:217-231.
4. Rosenfeld, A, Davis, LS. Image segmentation and image models. Proc. IEEE 67:764-772.

5. Zhang, YJ. A Survey on Evaluation Methods For Image Segmentation. Pattern Recognition 29:1335.
6. Abdou, IE, Pratt, K. Quantitative design and evaluation of enhancement/thresholding edge detectors. Proc. IEEE 67:753-763.
7. Ramesh, V, Haralick, RM. Performance Characterization of edge detectors. SPIE-Applications of Artificial Intelligence 1708:252-266.
8. Zhu, Q. Efficient evaluations of edge connectivity and width uniformity. Image and Vision Computing 14:21-34.
9. Kitchen, L, Rosenfeld, A. Edge evaluation using local edge coherence. IEEE Trans. Systems, Man and Cybernetics 11:597-605.
10. Palmer, PL, Dabis, H, Kittler, J. A performance measure for boundary detection algorithms. Computer Vision and Image Understanding 63:476-494.
11. Cho, K, Meer, P, Cabrera, J. Quantitative evaluation of performance through bootstrapping: Edge detection. IEEE Int'l Symp. Computer Vision491-496.
12. Heath, MD, Sarkar, S, Sanocki, T, Bowyer, KW. Comparison of Edge Detectors. Computer Vision and Image Understanding 69:38-54.
13. Fram, JR, Deutsch, ES. On the quantitative evaluation of edge detection schemes and their comparison with human performance. IEEE transactions on computers c-24:616-628.
14. Bryant, DJ, Boulding, DW. Evaluation of edge operators using relative and absolute grading. Proc. IEEE Computer society conf. pattern recognition and image processing138-145.
15. Strickland, RN, Cheng, DK. Adaptable edge quality metric. Optical Eng. 32:944-951.
16. Jiang, XY, Hoover, A, Jean-Baptiste, G, Bowyer, K. A methodology for evaluating edge detection techniques for range images. Proc. Asian conf. computer vision415-419.
17. Kanungo, T, Jaisimha, MY, Palmer, J, Haralick, RM. A methodology for quantitative performance evaluation of detection algorithms. IEEE Transactions on Image Processing 4:1667-1674.
18. Weszka, JS, Rosenfeld, A. Threshold evaluation techniques. IEEE Transaction on Systems, Man, and Cybernetics SMC-8:622-629.
19. Zhang, W, Yoshida, H, Nishikawa, RM, Doi, K. Optimally weighted wavelet transform based on supervised training for detection of microcalcifications in digital mammograms. Medical Physics 25:949-956.
20. Valverde, FL , Guil, N, Munoz, J. Bayesian approach based on geometrical features for validation and tuning of solution in deformable models. Iberamia 2002. LNCS Springer865-874.
21. Cheung, K, Yeung, D, Chin, R. On deformable models for visual pattern recognition. Pattern Recognition 35:1507-1526.
22. Kass, M, Witkin, A, Terzopoulos, D. Snakes : Active Contour Models. International Journal of Computer Vision321-331.
23. Amini, AA, Weymouth, TE, Jain, RC. Using dynamic programming for solving variational problems in vision. IEEE Transactions on Patterns Analysis and Machine Intelligence 12:855-866.
24. Williams, DJ, Shah, M. A fast algorithm for active Contours and Curvature Estimation. Computer Vision, Graphics and Image Processing 55:14-26.
25. Valverde, FL, Guil, N, Munoz, J. A deformable model for image segmentation in noisy medical images. IEEE International Conference on Image Processing, ICIP 200182-85.
26. Valverde, FL, Guil, N, Munoz, J. Segmentation of vessels from mammograms using a deformable model. Computer Methods and Programs in Biomedicine (2004) Vol 35: 233-247.

Structural Recognition with Kernelized Softassign

Miguel Angel Lozano and Francisco Escolano

Robot Vision Group,
Departamento de Ciencia de la Computación e Inteligencia Artificial,
Universidad de Alicante, Spain
{malozano, sco}@dccia.ua.es
http://rvg.ua.es

Abstract. In this paper we address the problem of graph matching and graph classification through a kernelized version of the classical Softassign method. Our previous experiments with random-generated graphs have suggested that weighting the Softassign quadratic cost function with distributional information coming from kernel computations on graphs yields a slower decay of matching performance with increasing graph corruption. Here, we test this approach in the context of automatically building graph prototypes and classifying graphs in terms of the distance to the closer prototype. In all cases we use unweighted graphs coming from real images and having sizes from 30 to 140 nodes. Our results show that this approach, consisting on applying graph kernel engineering to matching problems, has a practical use for image classification in terms of pure structural information.

1 Introduction

Graphs allow us to encode the significant structural/relational information of patterns yielding useful invariance for object recognition [1]. Given an input data graph representing an object, graph-based recognition is usually posed in terms of finding the most similar (structurally compatible) model graph stored in a database, even if there is a significant level of distortion between both graphs. There is a wide literature on *graph-registration* approaches: graph-matching [6][24][12], maximum clique finding [17], graph-edit distance minimization [19][2], and so on. However, the other side of the graph-based recognition coin, namely *graph categorization or graph clustering,* which must be addressed in order to yield efficient comparisons after having obtained useful abstractions, has been only addressed recently [9][13][20]. The latter tasks are even harder when only structural information is considered, that is, when neither node nor edge attributes are available to constrain the ambiguity arising in registration and categorization. We have recently proposed an approach to address both tasks in a coupled manner [14]. In this latter paper, registration is performed by a population-based version of the Softassign graph-matching algorithm [6] named Comb search which was originally proposed to solve MRF-labeling problems [11]. On the other hand, graph clustering was solved by adapting to the domain of graphs the Asymmetric Clustering Model (ACM) [7] [18] which yields good graph prototypes and the optimal number of classes provided that registration is good enough. However, in the pure structural case Comb search only provided acceptable matching results for graphs of 15 – 20 nodes. These poor results lead us to investigate and propose more robust, and hence powerful, versions of Softassign. We have recently *kernelized* the Softassign method [15] which consists on transforming the original matching problem between two non-attributed graphs into a matching problem between attributed ones and then use these attributes to weight the original quadratic cost function. The practical effect of this weighting is that it yields a good characterization of the local structure, which in turn helps to choose the proper attractor in a context of high ambiguity. We have reported matching results comparable to the use of more complex non-quadratic cost functions [5]. The term *kernelized* comes from the fact that we extract good attributes for the nodes of the non-attributed graphs by exploiting recent theoretical results in spectral graph theory [3]: Kondor and Lafferty [10] (see also [8] for a survey on kernels for structures like strings, trees and graphs) transferred to the domain of graphs the well-known concept of *kernel* defined for the domain of vectors [4][21][16] and they propose the diffusion kernels for graphs (the discrete version of Gaussian kernels) in order to measure the similarity between pairs of vertices of the same graph. More recently, Smola and Kondor have proposed a wider class of kernels on graphs, known as *regularization kernels* [23], which rely on studying the role of the Laplacian of a graph for regularization/smoothing. In section 2 we briefly review the fundamentals of regularized kernels and then we describe the kernelized version of Softassign. In section 3 we present our experiments both for graph registration and graph classification and outline our conclusions.

2 Kernelized Softassign

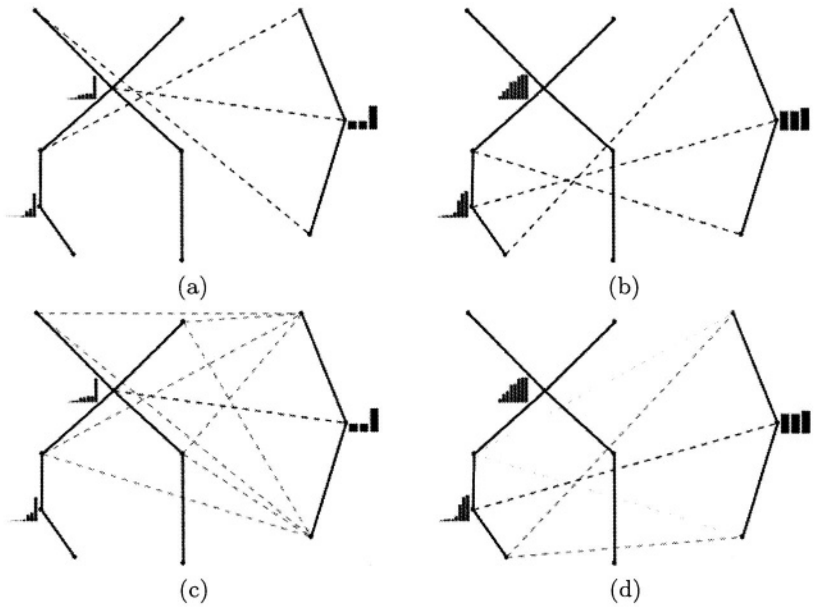
Given a undirected and unweighted graph $G = (V, E)$ with vertex-set V of size m, and edge-set $E = \{(i,j)|(i,j) \in V \times V, i \neq j\}$, its adjacency matrix and degree matrix are respectively defined as

$$A_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad D_{ij} = \begin{cases} \sum_{k=1}^{m} A_{ik} & \text{if } i = j \\ 0 & \text{otherwise} . \end{cases}$$

Then, the Laplacian of G, $L = D - A$, and its degree-normalized version, $\tilde{L} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}} = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ or Normalized Laplacian [3], are defined as

$$L_{ij} = \begin{cases} -1 & \text{if } (i,j) \in E \\ D_{ii} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad \tilde{L}_{ij} = \begin{cases} -D_{ii}^{-\frac{1}{2}} D_{jj}^{-\frac{1}{2}} & \text{if } (i,j) \in E \\ 1 & \text{if } i = j \text{ and } D_{ii} \neq 0 \\ 0 & \text{otherwise} . \end{cases}$$

The connection of the latter Laplacian matrices with regularization theory comes from spectral analysis, because a smoothness operator in Fourier space can



$$K_X^{Dif} = \begin{bmatrix} .474 & \mathbf{.190} & .082 & .106 & .083 & .026 & .032 & .007 \\ \mathbf{.190} & .177 & \mathbf{.133} & \mathbf{.190} & \mathbf{.138} & .064 & .083 & .026 \\ .082 & \mathbf{.133} & .279 & .082 & .064 & \mathbf{.214} & .026 & .120 \\ .106 & \mathbf{.190} & .082 & .474 & .083 & .026 & .032 & .007 \\ .083 & \mathbf{.138} & .064 & .083 & .306 & .021 & \mathbf{.299} & .006 \\ .026 & .064 & \mathbf{.214} & .026 & .021 & .336 & .006 & \mathbf{.308} \\ .032 & .083 & .026 & .032 & \mathbf{.299} & .006 & .521 & .001 \\ .007 & .026 & .120 & .007 & .006 & \mathbf{.308} & .001 & .524 \end{bmatrix} \quad K_Y^{Dif} = \begin{bmatrix} .526 & \mathbf{.317} & .158 \\ \mathbf{.317} & .366 & \mathbf{.317} \\ .158 & \mathbf{.317} & .526 \end{bmatrix}$$

$$K_X^{Reg} = \begin{bmatrix} .539 & \mathbf{.155} & .029 & .039 & .031 & .008 & .011 & .003 \\ \mathbf{.155} & .621 & \mathbf{.118} & \mathbf{.155} & \mathbf{.125} & .034 & .044 & .012 \\ .029 & \mathbf{.118} & .561 & .029 & .024 & \mathbf{.160} & .008 & .057 \\ .039 & \mathbf{.155} & .029 & .539 & .031 & .008 & .011 & .003 \\ .031 & \mathbf{.125} & .024 & .031 & .597 & .007 & \mathbf{.211} & .002 \\ .008 & .034 & \mathbf{.160} & .008 & .007 & .617 & .002 & \mathbf{.218} \\ .011 & .044 & .008 & .011 & \mathbf{.211} & .002 & .575 & .001 \\ .003 & .012 & .057 & .003 & .002 & \mathbf{.218} & .001 & .577 \end{bmatrix} \quad K_Y^{Reg} = \begin{bmatrix} .583 & \mathbf{.236} & .083 \\ \mathbf{.236} & .667 & \mathbf{.236} \\ .083 & \mathbf{.236} & .583 \end{bmatrix}$$

(e)

Fig. 1. Incidence of the choice of kernel in matching results. Matching results with the regularized Laplacian kernel: after (a) and before (c) performing the clean-up heuristic. Results with the diffusion kernel: (b) and (d). Dotted lines are preferred matches (the darker the stronger). Matrices associated to the kernels in each case (e)

be built by multiplying the Fourier transform by a penalizing function increasing in frequency. Denoting by $\mathcal{L}$ both L and $\tilde{L}$, and by $\{\lambda_i, \phi_i\}$ the eigenvalues and eigenvectors of $\mathcal{L}$, and being $r(\lambda_i)$ a monotone increasing function, it turns out that $r^{-1}(\lambda)$, the inverse of such a function, is the Fourier transform of the associated kernel in the continuous case, and the discrete regularization kernel K is the inverse (or the pseudo-inverse if necessary) of the so called *regularization matrix* $r(\mathcal{L})$. Then we have that

$$K = r^{-1}(\mathcal{L}) \text{ where } r^{-1}(\mathcal{L}) = \sum_{i=1}^{m} r^{-1}(\lambda_i)\phi_i\phi_i^T , \tag{1}$$

and $0^{-1} = 0$. The latter relation ensures the positive semi-definite condition which is necessary for K to be a kernel. In Table 1 we show several penalization functions and their associated regularization kernels. Particularly, the diffusion kernel relies on *matrix exponentiation* but not on componentwise exponentiation. For this kernel we have

$$K = e^{-\beta\mathcal{L}} = (\sum_{i=1}^{m} e^{\beta\lambda_i}\phi_i\phi_i^T)^{-1} = \sum_{i=1}^{m} e^{-\beta\lambda_i}\phi_i\phi_i^T .$$

In the general case, the relation $K = r^{-1}(\mathcal{L})$ is derived from the fact that given a regularization operator, for instance $M = r(\mathcal{L})$, the matrix K must satisfy the *self-consistency condition* $KMK = K$ to be a kernel, and therefore $K = M^{-1}$ or equal to the pseudo-inverse if M is not invertible. Furthermore, it can be proved [23] that such regularization operator defines a reproducing kernel Hilbert space whose kernel is $K = M^{-1}$.

What is important for graph matching is the fact that graph kernels, in general, and regularization kernels, in particular, define a similarity measure between vertices in the same graph. Then, it seems reasonable to assume that if two vertices match, the relative similarity between each of them and the rest of vertices in each graph should be compatible. In the Softassign formulation, a feasible solution to the graph matching problem between G_X and G_Y, with adjacency matrices X_{ab}, and Y_{ij}, is encoded by a matrix M of size $m \times n$, being $m = |V_X|$ and $n = |V_Y|$. Following the Gold and Rangarajan formulation we are interested in finding the feasible solution M that minimizes the following cost function,

$$F(M) = -\frac{1}{2}\sum_{a=1}^{m}\sum_{i=1}^{n}\sum_{b=1}^{m}\sum_{j=1}^{n} M_{ai}M_{bj}C_{aibj} , \tag{2}$$

where typically $C_{aibj} = X_{ab}Y_{ij}$. A simple way of *kernelizing* the latter energy function is to redefine C_{aibj} relying on H^{K_X} and H^{K_Y}, the entropies of the probability distributions associated to the vertices and induced by K_X and K_Y:

$$C^K_{aibj} = X_{ab}Y_{ij} \exp -[(H_a^{K_X} - H_i^{K_Y})^2 + (H_b^{K_X} - H_j^{K_Y})^2] , \tag{3}$$

Given a pair of vertices, for instance a, b of graph G_X the kernel K^X induces the following probability distribution

$$p_{ab}^X = K_{ab}^X(\sum_{c=1}^{m} K_{ac}^X)^{-1} \text{ and } H_a^{K_X} = \sum_{b=1}^{m} p_{ab}^X \log p_{ab}^X ,$$

and the same holds for p_{ij}^Y and $H_i^{K_Y}$ in case of vertices i, j of G_Y. We use entropy because building attributes in the properties of distributions yields more robustness than building them in the crude values of the kernels. This yields a definition of C_{aibj}^K ensuring that $C_{aibj}^K \leq C_{aibj}$, being the equality only verified when nodes a and i have similar entropies, and the same for nodes b and j. In practice, this weights the rectangles in such a way that rectangles with compatible entropies in their opposite vertices are preferred, and otherwise they are underweighted and do not attract the continuation process.

To see intuitively how the kernelized version works, in Fig. 1 we show the different matchings preferred by the regularized Laplacian kernel and the diffusion one after performing the clean-up heuristic (solving for ambiguity after the continuation process). In some vertices we show the probability distributions induced by each kernel. In this case, the diffusion kernel yields the more coherent matching in terms of structural subgraph compatibility.

Table 1. Penalization Functions and Regularization Kernels

$r(\lambda)$	$K = r^{-1}(\mathcal{L})$	Name
$1 + \beta\lambda$	$(I + \beta\mathcal{L})^{-1}$	Regularized Laplacian
$e^{\beta\lambda}$	$e^{-\beta\mathcal{L}}$	Diffusion Process
$(a - \lambda)^{-p}$	$(aI - \mathcal{L})^p$	p–step Random Walk
$(\cos \lambda\pi/4)^{-1}$	$\cos \mathcal{L}\pi/4$	Inverse Cosine

3 Experimental Results and Discussion

The key point of this paper is to show that the kernelized Softassign described above is a good choice for graph registration, and hence for graph clustering, when we have graphs coming from real-world images. We have considered the CMU, MOV1 and Chalet sequences, which are associated to observing different objects (houses) from a smooth range of viewpoints (10 frames). The CMU sequence has graphs of 30 vertices on average, with an averaged corruption of 10% between consecutive frames. The MOV1 sequence is very sparse (has graphs with 40 to 113 vertices) and the Chalet sequence has graphs of 131 –140 vertices.

In the upper row of Fig. 2, we show very good matching results between graphs associated to consecutive frames *(graph tracking)* of the CMU sequences (almost 100% of success). In the middle row we show also good tracking results even for the Chalet sequence which have very big graphs. Finally, in the bottom row we show good matching results between distant frames of the CMU sequence (left) whereas the results are not so good when trying to match a graph from CMU to a graph from Chalet. In this latter case, the kernelized Softassign only yields good local matches (subgraph to subgraph).

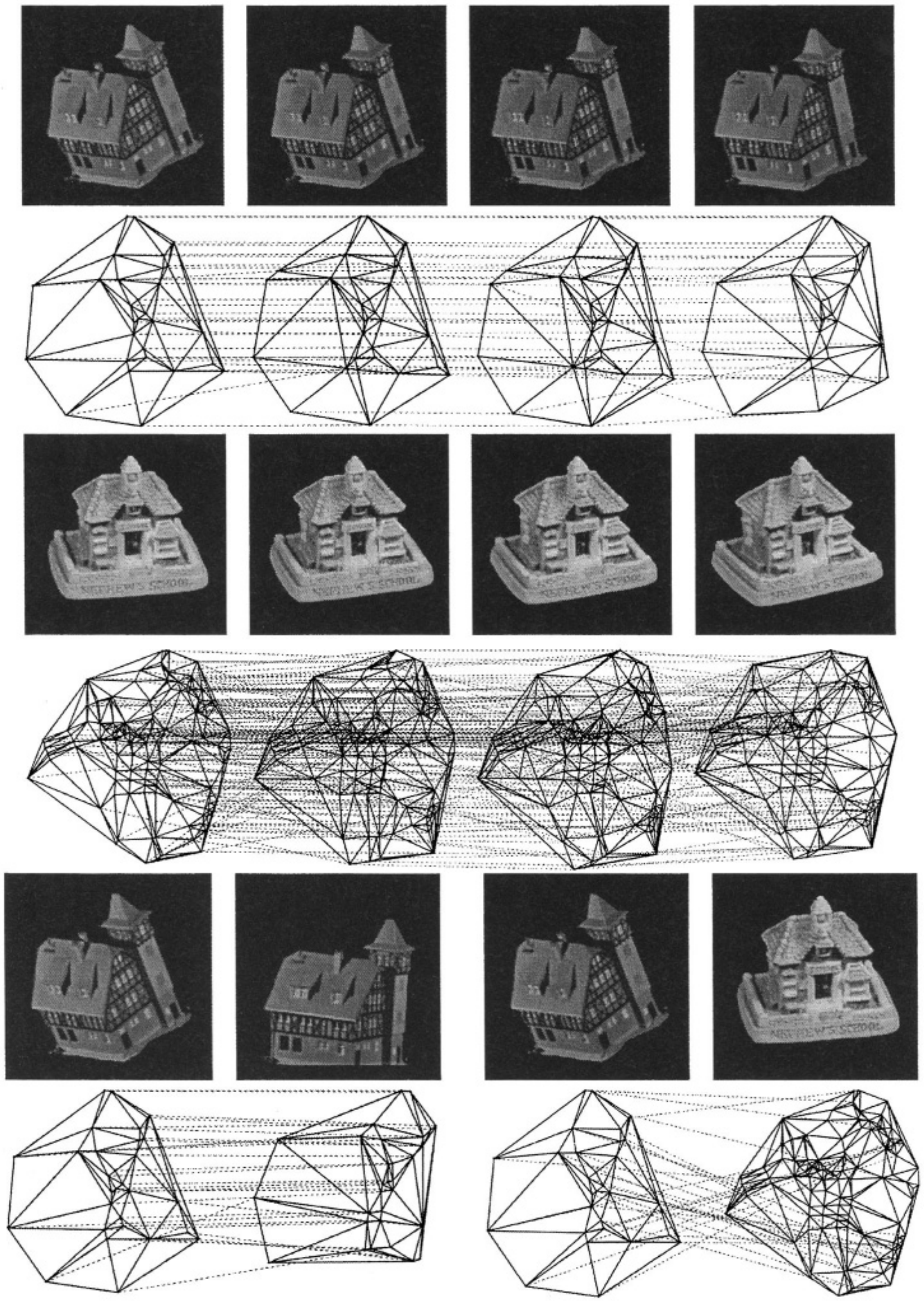


Fig. 2. Matching results for the CMU and Chalet sequences

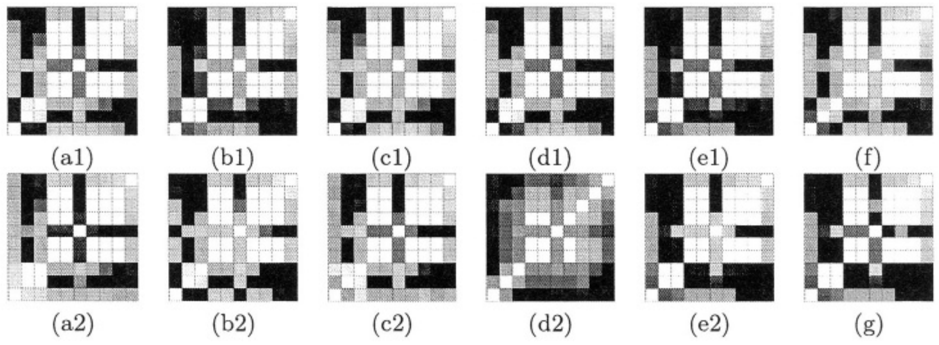


Fig. 3. Evaluating kernel performance for the CMU sequence: (a1,a2) diffusion kernel, (b1,b2) diffusion kernel with β normalized by the number of nodes, (c1,c2) regularized Laplacian, (d1,d2) 2–step, (e1,e2) cos, (f) Softassign driven by degree similarity, and (g) classical Softassign. Top row $\tilde{L}$, bottom row L

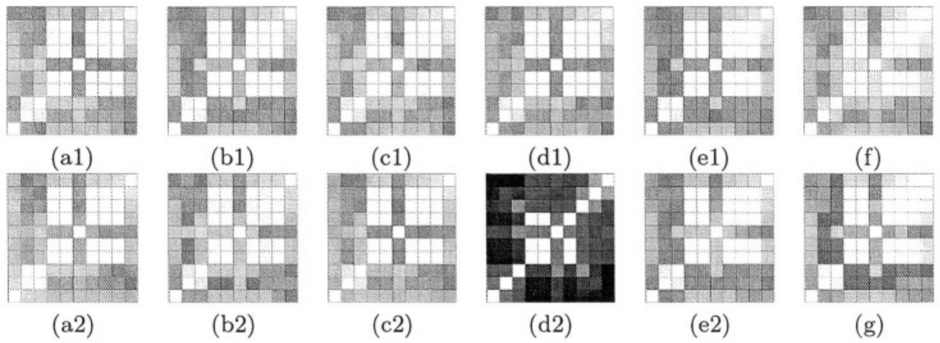


Fig. 4. Evaluating kernel performance from the optimal cost

In all the experiments above we have used the diffusion kernel with L because such a kernel provides the best performance (Fig. 3): Knowing beforehand the correct matchings we have evaluated the rate of success for all the pairings between graphs of the CMU sequence; each cell corresponds to a pair of graphs (in each *confusion matrix* white indicates 100% of success and black indicates 0% of success). We have tested all kernels both with L and $\tilde{L}$; the diffusion kernel for L (bottom row and first column) obtains more white cells and also the number of successful matchings degrades gracefully with the distance between the frames. On the other hand, when we do not know the correct solution beforehand we need to estimate the normalized degree of similarity between the graphs. This is why in Fig. 4 we have showed compatible results with that of Fig. 3 but in terms of the distance between two graphs (white represents high cost or low distance whereas black represents the opposite).

Once we have used the best kernel for graph registration we address the problem of graph classification. In Fig. 5 we show the prototype for graphs in the

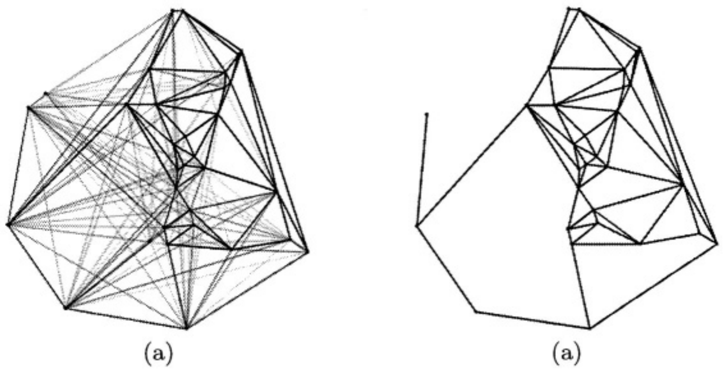


Fig. 5. Prototypical graph for the CMU sequence before (a) and after (b) thresholding nodes and edges with probabilities below 0.5

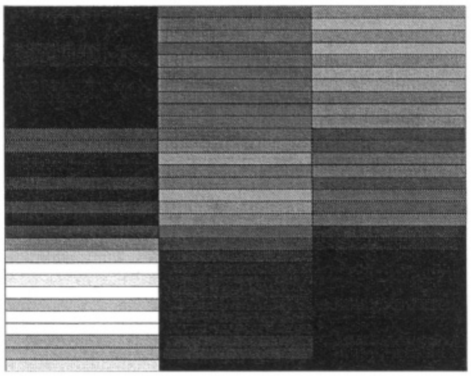


Fig. 6. Distances between graphs and prototypes in sequences CMU, MOV1 and Chalet

CMU sequence after the incremental fusion process described in [14]. We have also obtained the prototypes for the other two sequences, MOV1 and Chalet. Finally, in Fig. 6 we showed the minimal distances (in terms of the normalized optimal cost) between all graphs (first 10 bottom rows for CMU, next 10 middle rows for MOV1, and next 10 top rows for Chalet) and the three prototypes (first column for CMU, second for MOV1, and third for Chalet). Considering the CMU prototype, graphs are well classified; for MOV1 and Chalet, the results are acceptable if one considers the high level of sparseness of these classes and the high number of vertices (in the case of the Chalet sequence). Visually, the blocks in the diagonal are pretty seen in white.

As a conclusion, with the kernelized Softassign we can solve the graph-registration problem even for very complex graphs which in turns implies good classifications provided after prototype estimation.

Acknowledgements

This work was partially supported by grant *TIC*2002 – 02792 funded by *Ministerio de Ciencia y Tecnología* of the Spanish Government and by *FEDER.*

References

1. Bunke,H.: Recent Developments in Graph Matching. In Proceedings of the International Conference on Pattern Recognition (ICPR'00)- Vol.2 (2000) 2117–2124
2. Bunke,H.: Error Correcting Graph Matching: On the Influence of the Underlying Cost Function. IEEE Transactions on Pattern Analysis and Machine Intelligence **21** (9) (1999) 917–922
3. Chung, F.R.K.: Spectral Graph Theory. Conference Board of the Mathematical Sciences (CBMS) **92.** American Mathematical Society (1997)
4. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines Cambridge University Press (2000)
5. Finch, A.M., Wilson, R.C., Hancock, E.: An Energy Function and Continuous Edit Process for Graph Matching. Neural Computation, **10** (7) (1998) 1873-1894
6. Gold, S., Rangarajan, A.: A Graduated Assignment Algorithm for Graph Matching. IEEE Transactions on Pattern Analysis and Machine Intelligence **18** (4) (1996) 377–388
7. Hofmann, T., Puzicha, J.: Statistical Models for Co-occurrence Data. MIT AI-Memo 1625 Cambridge, MA (1998)
8. Gärtner: A Survey of Kernels for Structured Data. ACM SIGKDD Explorations Newsletter **5**(1) (2003) 49–58
9. Jiang, X., Münger, A., Bunke, H.: On Median Graphs: Properties, Algorithms, and Applications. IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 23, No. 10 (2001) 1144-1151
10. Kondor, R.I., Lafferty, J.: Diffusion Kernels on Graphs and other Discrete Input Spaces. In: Sammut, C., and Hoffmann, A. G. (eds) Machine Learning, Proceedings of the Nineteenth International Conference (ICML 2002). Morgan Kaufmann (2002) 315–322
11. Li, S.Z.: Toward Global Solution to MAP Image Estimation: Using Common Structure of Local Solutions. In: Pelillo, M., Hancock, E.R.(eds.)proceedings of the 1st Internatinal Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR 1997). Lecture Notes in Computer Science, Vol. 1223. Springer-Verlag, Berlin Heidelberg New York (1997) 361-374.
12. Luo, B., Hancock, E.R.: Structural Graph Matching Using the EM Algorithm and Singular Value Decomposition. IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 23, No. 10 (2001) 1120-1136.
13. Luo,B., Wilson, R.C, Hancock,E.R.: A Spectral Approach to Learning Structural Variations in Graphs. In: Crowley,J.L., Piater, J.H., Vincze,M., Paletta, L.(Eds.) Proceedings of Computer Vision Systems, Third International Conference (ICVS 2003) Proceedings. Lecture Notes in Computer Science 2626 Springer (2003) 407–417
14. Lozano,M.A., Escolano, F.: EM Algorithm for Clustering an Ensemble of Graphs with Comb Matching. In: Rangarajan, A., Figueiredo, M., Zerubia, J. (Eds.) Proceedings of the 4th International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR 2003). Lecture Notes in Computer Science 2683 (2003) 52–67

15. Lozano, M.A., Escolano, F.: A Significant Improvement of Softassing with Diffusion Kernels. In Proceedings of the IAPR International Workshop on Syntactical and Structural Pattern Recognition SSPR 2004. Lecture Notes in Computer Science (2004) (accepted for publication)
16. Müller, K.-R., Mika, S., Räshc, Tsuda, K., Schölkopf, B.: An Introduction to Kernel-based Learning Algorithms. IEEE Transactions on Neural Networks, **12**(2) (2001) 181–201.
17. Pelillo, M.: Replicator Equations, Maximal Cliques, and Graph Isomorphism. Neural Computation **11** (1999) 1933–1955
18. Puzicha, J.: Histogram Clustering for Unsupervised Segmentation and Image Retrieval. Pattern Recognition Letters, 20, (1999) 899-909.
19. Sanfeliu,A., Fu, K.S.: A Distance Measure Between Attributed Relational Graphs for Pattern Recognition. IEEE Transactions on Systems, Man, and Cybernetics **13** (1983) 353–362
20. Serratosa, F., Alquézar, R., Sanfeliu, A.: Function-described graphs for modelling objects represented by sets of attributed graphs, Pattern Recognition, Vol. 23, No. 3 (2003) 781-798
21. Schölkopf, B., Smola, A.: Learning with Kernels. MIT Press (2002).
22. Smola, A., Schölkopf, B., Müller, K.-R.: The Connection between Regularization Operators and Support Vector Kernels. Neural Networks **11** (1998) 637–649
23. Smola, A., Kondor, R.I.: Kernels and Regularization on Graphs. In: Schölkopf,B., and Warmuth, M. K. (eds) Computational Learning Theory and Kernel Machines, 16th Annual Conference on Computational Learning Theory and 7th Kernel Workshop, COLT/Kernel 2003. Lecture Notes in Computer Science. Springer. Vol. **2777** (2003) 144–158
24. Wilson, R.C., Hancock, E.R.:Structual Matching by Discrete Relaxation. IEEE Transactions on Pattern Analysis and Machine Intelligence **19** (6) (1997) 634–648

Kernel Based Method for Segmentation and Modeling of Magnetic Resonance Images*

Cristina García and José Alí Moreno

Laboratorio de Computatión Emergente,
Facultades de Ciencias e Ingeniería,
Universidad Central de Venezuela
{cgarcia, jose}@neurona.ciens.ucv.ve
http://neurona.ciens.ucv.ve/laboratorio

Abstract. In this paper we propose a method for segmenting structures in Magnetic Resonance Images (MRI) using the well known kernel Adatron algorithm. The method allows the segmentation in feature space of 2D structures present in each slice of a set of MRI data. The resulting 2D segmented regions are used as input of a kernel SVM algorithm to produce a three dimensional model of the object. This procedure enables the physician the visualization of the complete 'real' structure, difficult to perceive from the individual 2D slices of the original MRI data. The representation of the 3D model is obtained as an implicit function, allowing volume, distances and related measurements. The methodology is applied to the segmentation and modeling of a specific structure: a brain tumor from a real data set of a human brain.

1 Introduction

MRI is one of the most used techniques to register the brain. The images produced, 2D slices along the brain volume, contain many details, including gray matter, white matter and intracranial cerebrospinal fluid. The segmentation of the brain from MRI scans has important applications in neuroimaging: for the visualization and quantification of the cortex, for volume calculations that can be related to measures of disease severity, to quantify morphological differences and changes due to neurological disorders and so forth [1]. It is clear then why so much effort is dedicated to construct methods capable of producing accurate segmentations of brain images [2].

Usually, MRI automated segmentations are done under an Expectation Maximization framework where atlas information of anatomical structures is used either as apriori probability or as a basis to deform the shape. This procedure sometimes increases the risk of systematical biases [3]. Other method calculates probability maps by adjusting different functions to model the intensity probability distribution of the voxels in various combination of tissues and implementing

* This research was supported by the Fondo National de Ciencia, Tecnología e Innovatión (FONACYT) under project G-97000651.

a fuzzy classification scheme, being its main disadvantage the relative complexity of the algorithm [1].

In this paper we propose an alternative segmentation method using a learning machine algorithm: the Adatron [4]. The issue we are trying to address is if it is possible to build a segmentation of the brain images without using apriori information represented by theoretic probability models of tissue distributions. We want to obtain a segmenting machine guided only by local information extracted from the image. To achieve a segmentation based approximately on anatomical dependencies, a vector composed by the pixel to be classified and the pixels in its neighborhood is constructed. The classification machine is trained on those instances. To gain representation capacity the Adatron algorithm is combined with the powerful methodology of Kernel functions [5]. This combination enhances the performance of the linear machine enabling the construction of general non-linear decision boundaries between classes.

To illustrate one of the possible applications of the segmentation method proposed, a specific 2D structure is extracted in a set of MRI data. The segmentation produced is sampled and points are used to synthesize a model of its 3D appearance. In order to construct a 3D model, the Support Vector Machine (SVM) formalism is used to calculate a hypersphere of minimum radius enclosing all data points. This is accomplished in an arbitrary feature space induced through a Kernel function [6]. The approach follows a clustering algorithm suggested by Ben-Hur et. al. [7] and has been shown to be suitable for object modelling [8], with enough flexibility to represent objects in medical images [9]. Thus, instead of multiple partial slices, the whole object in its volumetric representation could be given to the physician. The more natural 3D perspective improves the visualization of the object and as a consequence, diagnosis and therapy.

The paper is organized as follows: section 2 explains the Kernel-Adatron segmentation procedure, section 3 the SVM approach to 3D modeling, section 4 presents experimental results obtained and finally section 5 includes the conclusions derived and the perspectives for future work.

2 Segmentation of MRI Data

The segmentation of brain MR images is a challenging problem. The different structures composing each image and their ramified topology contribute to the complex segmentation scenario. Manual delineation by anatomists is still a common practice to obtain high quality segmentations. Improvements in automatic segmentation are recently being produced by incorporating previous knowledge of the anatomical structures involved [3]. Nevertheless, using apriori probabilities has the consequence of producing systematical biases and can not be effectively applied when the patient has a special condition produced by a neurological disorder or similar disease.

In this work we study an alternate methodology for building brain MRI segmentations based on local information present in the image. In doing so, we expect to reduce the biases and provide a framework able to handle even non-typical conditions in individuals. The following two sections briefly describe the proposed segmenting machine.

2.1 Neighborhood

Spatial, intensity and shape distributions of the anatomical structures of interest are usually incorporated as prior knowledge in the segmentation task through probability models. As an alternative to correlate those factors, it seems natural to group nearby pixels into neighborhoods and investigate if local variations of the pixel intensities, allow the identification of underlying anatomical structures. To this end, two types of neighborhoods are used: von Neumann and Moore, whose names are borrowed from the Cellular Automata literature [10]. Considering a two dimensional lattice, the following definitions apply:

– von Neumann Neighborhood: the reference cell (center) plus the cells above and below, right and left. The radius of this definition is 1, as only the next layer is considered (Fig. 1 - Left). An extension of the von Neumann neighborhood can be constructed increasing the radius, and consequently considering more layers (Fig. 1 - Center).
– Moore Neighborhood: this is an enlargement of the von Neumann neighbourhood containing the diagonal cells too. In Fig. 1 - Right, the case for radius r=1 is shown. Larger radius extend Moore neighborhoods to next adjacent cells, analogously to the von Neumann extension.

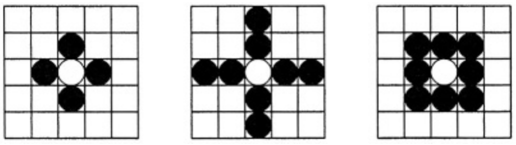


Fig. 1. Neighborhoods. Left: von Neumann, Center: von Neumann Extension, Right: Moore

It must be noted that the von Neumann neighborhood needs less components to cover a specific radius than the Moore neighborhood. For example, a von Neumann neighborhood with radius 6, uses 25 components, while this size correspond to a Moore neighborhood with radius 2. Since a larger radius includes greater dispersion in intensities for boundary regions and less variety for interior pixels, it provides a better distinction between different structures and/or shapes. Thus, the use of von Neumann neighborhoods seem adequate to represent the desired relations between intensity distributions. Furthermore, as different anatomical structures are characterized by different intensities it is logical to assume that each sort of tissue will correspond to a particular neighborhood imprint.

Although this segmentation method pretends to infer anatomical structure it is worth to take into account that it is not a model driven technique, there are not anatomical constraints involved nor established previously. Instead, instances of vectors composed by the pixel to be classified and its neighboring pixels are constructed. The classification machine is trained on those instances, resulting in a segmentation based approximately on anatomical dependencies.

2.2 Kernel Adatron Algorithm

One of the 'meta' objectives in solving a classification problem is to produce a high quality decision function. In the learning machine literature [6] this solution is known as an optimal separating hyperplane, in the statistical physics literature [4] as the solution with maximal stability. This problem has been exhaustively studied in the literature [6,4], thus in what follows, only main definitions and results are included.

Consider a training set $S = |(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)|, \boldsymbol{x}_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$ where the $\boldsymbol{x}_i$ are called the training vectors in input space, with dimension d, and the y_i are their corresponding class labels. The objective of the classification is to construct from the finite data set S a good decision function $f(\boldsymbol{x})$ such that:

$$y_i = f(\boldsymbol{x}_i) \quad \forall (\boldsymbol{x}_i, y_i) \in S \tag{1}$$

The criteria for a solution $f(\boldsymbol{x})$ to be acceptable is that the functional margins of all training points be positive:

$$\gamma_i = y_i \; f(\boldsymbol{x}_i) \geq 0 \quad \forall (\boldsymbol{x}_i, y_i) \in S \tag{2}$$

In the case of a linear decision function this criterion can be extended to the geometrical margins:

$$\Gamma_i = \frac{\gamma_i}{||\boldsymbol{w}||} = \frac{y_i(\langle \boldsymbol{w} \cdot \boldsymbol{x}_i \rangle + b)}{||\boldsymbol{w}||} \geq 0 \quad \forall (\boldsymbol{x}_i, y_i) \in S \tag{3}$$

It is clear from relation (3) that the most stable solution should be the one with the greatest minimum geometrical margin. This is called the maximal margin classifier or perceptron with maximal stability. Since the definition of functional margin (2) is dependent on the scale of the weight vector it is common practice to choose a scale such that the minimum functional margin is unity. With this assumption the problem of obtaining the parameters of the maximal margin classifier can be stated as:

$$\text{Minimize } \frac{1}{2}||\boldsymbol{w}||^2 \tag{4}$$

$$\text{subject to } \gamma_i = y_i \; (\langle \boldsymbol{w} \cdot \boldsymbol{x}_i \rangle + b) \geq \; 1 \quad \forall (\boldsymbol{x}_i, y_i) \in S \tag{5}$$

Several computational procedures have been proposed to solve this optimization problem. In this work the algorithm applied to the classification task due to its simplicity is the Adatron.

Non linear generalizations of the maximal margin classifier can readily be made taking advantage of the kernel theory. Elaborations in this respect can be found in the literature [6,5]. Summarizing, it is shown that the form of the resulting kernel machine is:

$$f(\boldsymbol{x}) = \sum_{i}^{n} \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b \tag{6}$$

Where $K(\boldsymbol{x}_i, \boldsymbol{x})$ is the kernel function and the α_i are the Lagrange multipliers resulting from the solution of the optimization problem in its dual representation. In this work the Gaussian kernel is used throughout:

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(\frac{-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{\sigma^2}) \tag{7}$$

where σ stands for the width parameter of the Gaussian function.

3 Support Vector Kernel Modeling

The idea behind the application of the support vector formalism [6] to object modeling follows the SVM clustering method in [7]. This algorithm separates clusters of arbitrary geometrical shapes according to valleys in the underlying probability distribution. It is possible also to tightly enclose most of the training points in an unique cluster. The boundary generated can then be used as a representation of the cluster. In the particular case where the data points belong to an object that lives in 3D space, this contour can be used to represent the surface of the object [8]. Experiments have shown that very complicated objects as the ones encountered in medical images can be modeled as well with this procedure [9]. Details of the method can be found in references [7–9], the remaining of this section summarizes the main issues.

Let $\{x\}$ be a data-set of N points in an n-dimensional input space. Let ϕ be a nonlinear transformation from $\mathbb{R}^n$ to some high dimensional space, called feature space. The objective is to find the smallest hypersphere of radius R that encloses most of the points in feature space. If $\boldsymbol{a}$ denotes the center of the hypersphere, and slack variables ξ_i are introduced to account for outliers, the equation of the hypersphere can be written in terms of the Euclidean norm as:

$$\|\phi(x_i) - \boldsymbol{a}\|^2 \leq R^2 + \xi_i \quad i = 1, \ldots, N \tag{8}$$

$$\xi_i \geq 0 \tag{9}$$

The problem to solve is then:

$$\text{Minimize} \quad R^2 + C \sum_i \xi_i \tag{10}$$

$$\text{subject to} \quad \|\phi(x_i) - \boldsymbol{a}\|^2 \leq R^2 + \xi_i \tag{11}$$

$$\xi_i \geq 0, \quad i = 1, \ldots, N \tag{12}$$

where C represents a regularization constant that weights the conflicting objectives: maintain a low number of outliers vs. minimize the radius of the hypersphere.

To solve this optimization problem the Lagrangian formalism is applied. The optimization problem in its dual formulation can be stated as:

$$\theta = \sum_i \beta_i \langle \phi(x_i) \cdot \phi(x_i) \rangle - \sum_{i,j} \beta_i \beta_j \langle \phi(x_i) \cdot \phi(x_j) \rangle \tag{13}$$

This expression is given in terms of dual variables, the Lagrange multipliers β_i. The solution of the problem reduces to obtaining the set of Lagrange multipliers through the maximization of the dual formulation (13). This is computed using the Sequential Minimal Optimization (SMO) algorithm proposed by Platt [11].

The dot product operations of the feature space transformed points can be conveniently expressed through kernel functions. This formulation replaces the search for an expressive transformation with the application of an adequate kernel. Thus, explicit transformation of data is unnecessary with the consequence that the dimension of feature space is irrelevant. In this work a Gaussian kernel, defined in (7), is used also in the modeling stage.

Finally, it must be noted that the distance of a point x to the center of the hypersphere in feature space is given by:

$$R^2(x) = K(x,x) - 2\sum_j \beta_j K(x_j, x) + \sum_i \sum_j \beta_i \beta_j K(x_i, x_j)$$

This defines an implicit representation of the surface in feature space when the radius used is the radius of the hypersphere. Some of the training points result to be in the surface of the hypersphere, those are called support vectors (SV) [7]. Furthermore, the expression of the surface of the hypersphere that encloses all data points in feature space also defines the surface in 3D input space, as the implicit representation induced by the kernel preserves the topology of the input space. Thus, the surface of the 3D object can be described by the set:

$$\{x \mid R(x) = R\}, \text{ with } R = \{R(x_i) \mid x_i \text{ is SV}\} \tag{14}$$

4 Experimental Results

For experimentation one MRI data set of a human brain, composed of 200 slices of 483x472 pixels images, T2 volumes, with 0.45mm x 0.46mm x 0.48mm resolution was used. The images were segmented into background, exterior - combination of skull, muscle and skin -, intracranial cerebrospinal fluid and brain. Furthermore, as can be noted in the images included, the MRI correspond to a subject with abnormal tissue: a brain tumor. The tumor deforms the underlying structure almost in every slice of the brain MRI, even though it occupies approximately 37 slices (18.5%) of data set: from slice No. 98 to slice No. 134.

The data to train the Kernel-Adatron was extracted from slices 104 to 107. Small regions were sampled and manually labeled to one of the five classes: Background, Exterior, Cerebrospinal Fluid, Brain and Tumor. To approximate anatomical dependencies, configurations with von Neumann's neighborhood of radius 6 were built, resulting in patterns of 25 components as explained before (Sec. 2.1). The distribution of classes in training data is included in table 1.

The training of the Kernel-Adatron was conducted in a one against all basis (except background), obtaining 4 different non-linear decisions functions to recognize each kind of tissue. As tumor deforms great extension of the brain, samples labelled Tumor were merged with samples labelled Brain in all cases, excluding of course the case for tumor recognition. Table 2 shows the parameters of the classification functions calculated. The proportion of sample points per class used for training is not correlated with the distribution of SVs obtained. This is highly dependent on the samples of points used, in spite of this, the optimality of the Adatron assures reasonable good results. Fig. 2 shows an MRI slice and the segmentation in the corresponding structures: exterior, intracranial cerebrospinal fluid and brain. Analog results for slice No. 90 are displayed in Fig. 3.

The Adatron trained on the tumor tissue was used in the segmentation of slice No. 98 through slice No. 134 of the data set. A sample of 6253 points belonging to the tumor, according to the segmentation machine, was used to obtain a 3D representation of it. The model constructed with $\sigma^2 = 10$, uses 399 SVs. This sparsity is a common characteristic of the SVM approach. Three different views of the tumor model are presented in Fig. 4, and a comparison between the segmentation of MRI slice No. 120 produced by the Adatron and the correspondent cross section in the 3D model is included in Figure 5.

Table 1. Distribution of Classes in Training Data

Class	Region / Tissue	Number of Samples	Percentage
1	Background	25	1.0%
2	Exterior	427	17.2%
3	Cerebrospinal Fluid	401	16.1%
4	Brain	943	38.0%
5	Tumor	687	27.7%
	Total	2483	100%

Table 2. Kernel-Adatron Results for Segmentation

Class	σ^2	Number of SVs	SVs in class	SVs in the other classes
2	5000	816	359	457
3	5000	806	235	571
4	5000	808	249	559
5	5000	483	74	409

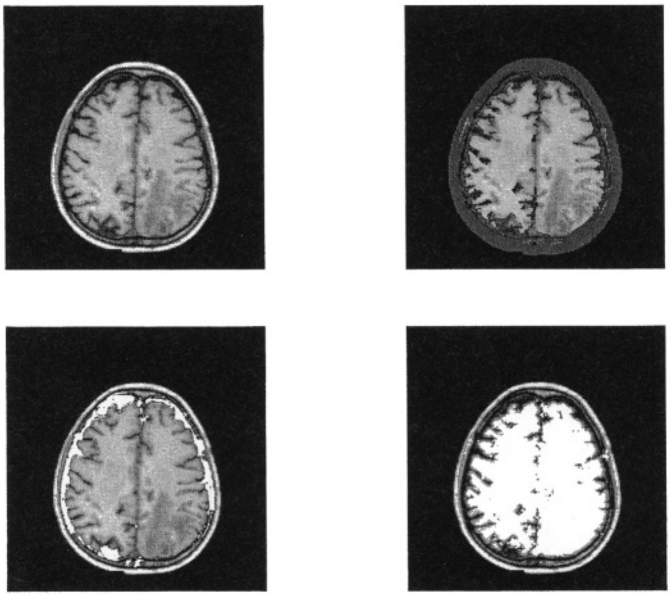


Fig. 2. Different Segmentations of Slice No. 50. Top Left: Original, Top Right: Exterior, Bottom Left: Cerebrospinal Fluid, Bottom Right: Brain

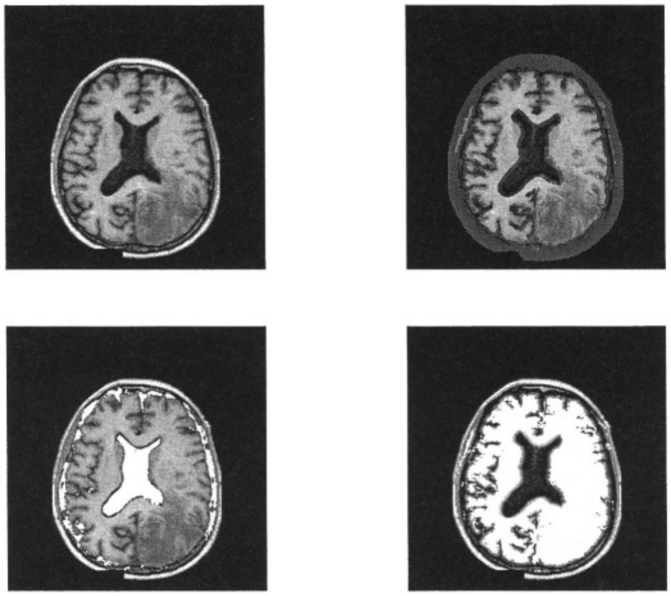


Fig. 3. Different Segmentations of Slice No. 90. Top Left: Original, Top Right: Exterior, Bottom Left: Cerebrospinal Fluid, Bottom Right: Brain

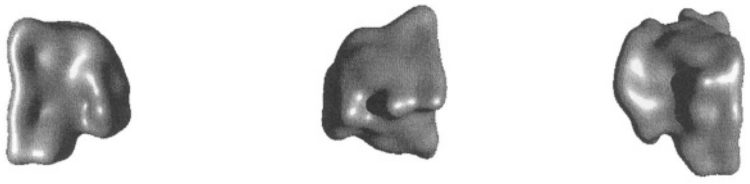


Fig. 4. 3D Modeled Tumor

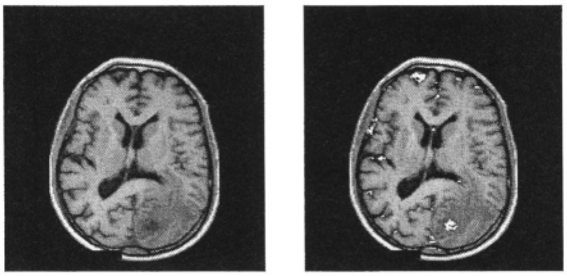


Fig. 5. Slice Segmented vs. Model Cross Section. Left: Original, Center: Segmented Tumor, Right: Cross Section of the 3D Modeled Tumor

5 Conclusions and Future Work

In this work a kernel based segmentation method for MRI brain images is proposed. It is shown that it is possible to obtain reasonable segmentations without any prior model assumption of tissue distribution. The configuration of the training patterns in terms of intensity values of the neighboring pixels enables an inference of the underlying anatomical structure. It is noteworthy that the information needed to produce the segmentations is very reduced. The power of the method lies on the generalization capacity of the applied kernel machines. Although these preliminary results are very promising, it is clear that further work must be conducted to enhance the segmentation procedure. There are several physiological restrictions unaccounted for: anatomically it is known that gray and white matter share a boundary, white matter is a singly convoluted structure and there is no isolated gray matter present. Those constraints could serve for validation purposes and eventually be introduced as part of an improved segmenting machine. Nevertheless it must be noted that in order to produce a proper model the method does not strictly require a high quality segmentation. Also the normally used conditions regarding the connectivity, convexity or closure of the structure under consideration may be relaxed. Thus, the main concern that still remains is finding a meaningful error measure, in general an open problem in 3D object modeling.

Acknowledgements

Gratitude is expressed to Dr. Miguel Martin for his advice and for allowing us the utilization of the MRI data set.

References

1. Lemieux, L., Hammers, A., Mackinnon, T., Liu, R.: Automatic segmentation of the brain and intracranial cerebrospinal fluid in t_1-weighted volume mri scans of the head, and its application to serial cerebral and intracranial volumetry. Magetic Resonance in Medicine **49** (2003) 872–884
2. Pham, D., Xu, C., Prince, J.: Current methods in medical image segmentation. Annual Review in Biomedical Engineering **2** (2000) 315–337
3. Pohl, K., Grimson, W., Bouix, S., Kikinis, R.: Anatomical guided segmentation with non-stationary tissue class distibutions in an expectation-maximization framework. IEEE International Symposium on Biomedical Imaging (2004) 81–84
4. Anlauf, J., Biehl, M.: The adatron - an adaptive perceptron algorithm. Europhysics Letters **10** (1989) 687–692
5. Friest, T., Campbell, C., Cristianini, N.: The kernel-adatron: A fast and simple learning procedure for support vector machines. In: Proceedings of the Fifteenth International Conference on Machine Learning. Morgan - Kaufmann, San Francisco, CA (1998)
6. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines. Cambridge University Press (2000)
7. Ben-Hur, A., Horn, D., Siegelmann, H., Vapnik, V.: A support vector method for clustering. International Conference on Pattern Recognition (2000)
8. Garcia, C., Moreno, J.: Application of learning machine methods to 3d object modeling. In Grijalbo, A., Clavijo, J., eds.: Advances in Artificial Inteligence - IBERAMIA 2002. Springer-Verlag, Berlin (2002) 453–462
9. Garcia, C., Moreno, J.: Application of support vector clustering to the visualization of medical images. IEEE International Symposium on Biomedical Imaging (2004) 1553–1556
10. Tofolli, T., Margolus, N.: Cellular Automata Machines. The MIT Press, Cambridge, Mass (1987)
11. Platt, J.: Fast training of support vector machines using sequential minimal optimization. In Scholkopf, B., Burges, J., Smola, A., eds.: Advances in Kernel Methods - Support Vector Learning. MIT Press, Cambridge, MA (1999) 185–208

Real-Valued Pattern Recall by Associative Memory

Humberto Sossa, Ricardo Barrón, and José L. Oropeza

Centro de Investigación en Computación-IPN,
Av. Juan de Dios Bátiz, esquina con Miguel Othón de Mendizábal,
Mexico City, 07738. Mexico

Abstract. In this note, we introduce an associative memory useful to recall real-valued patterns altered with mixed noise (additive and subtractive). Numerical and real examples are given to show the effectiveness of the proposal. Conditions under which the proposed memories are able to recall patterns either from the fundamental set of patterns and from distorted versions of them are also given.

1 Introduction

Humans are able to recall patterns with a great facility. Trying to implement this ability into a computer has been a major topic among the computer vision community. Several approaches have been proposed to give a solution to this important problem. One approach is based on the so-called neural networks. An associative memory (AM) is a kind of neural network. It is a device specially designed to recall patterns. An associative memory **M** can be viewed as an input-output system as follows: $\mathbf{x} \to \mathbf{M} \to \mathbf{y}$, with **x** and **y**, respectively the input and output patterns vectors. Each input vector forms an association with a corresponding output vector. An association between input pattern **x** and output pattern **y** is denoted as (x,y). For k integer and positive, the corresponding association will be denoted as $\left(\mathbf{x}^k, \mathbf{y}^k\right)$. The associative memory **M** is represented by a matrix whose ij-th component is m_{ij}. **M** is generated from a finite a priori set of known associations, known as the *fundamental set of associations,* or simply the *fundamental set* (FS). If ξ is an index, the fundamental set is represented as: $\left\{\left(\mathbf{x}^\xi, \mathbf{y}^\xi\right) \mid \xi = 1,2,\ldots,p\right\}$ with p the cardinality of the set. The patterns that form the fundamental set are called *fundamental patterns.* If it holds that $\mathbf{x}^\xi = \mathbf{y}^\xi \ \forall \xi \in \{1,2,\ldots p\}$, **M** is auto-associative, otherwise it is hetero-associative. A distorted version of a pattern **x** to be recuperated will be denoted as $\tilde{\mathbf{x}}$. If when feeding a distorted version of $\mathbf{x}^w$ with $w \in \{1,2,\ldots,p\}$ to an associative memory **M**, it happens that the output corresponds exactly to the associated pattern $\mathbf{y}^w$, we say that recalling is perfect. Several models for associative memories have emerged in the last 40 years. Refer for example to [1-11]. Works reported in [6], [8] and [11] are the closest foundation of the work reported in this paper. The main difference of our proposal with those described in [6], [8] and [11] is that ours can handle with mixed noise. In [6] the authors propose an initial solution by means of the so-called kernels. Its proposal works only with binary patterns, ours works with real-valued patterns. A method to find the desired set of kernel is however a very difficult task.

In this paper we introduce an associative memory useful to recall real-valued patterns when they appear altered by mixed noise. We give the formal conditions under which the proposed memories are able to perfectly recall a pattern of the FS set or when a corrupted version of one of them with mixed noise is presented as input to the memory. We present also several numerical and real examples where the new memories demonstrate their effectiveness.

2 Foundations

Let $P = \left[p_{ij}\right]_{m\times r}$ and $Q = \left[q_{ij}\right]_{r\times n}$ two matrices.

Definition 1. *The following two matrix operations are defined next defined:*

1. Operation $\Diamond_{\mathrm{A}}$: $P_{m\times r}\Diamond_{\mathrm{A}}Q_{r\times n} = \left[f_{ij}^{\mathrm{A}}\right]_{m\times n}$ where $f_{ij}^{\mathrm{A}} = \bigotimes_{k=1}^{r} \mathrm{A}\left(p_{ik}, q_{kj}\right)$.
2. Operation $\Diamond_{\mathrm{B}}$: $P_{m\times r}\Diamond_{\mathrm{B}}Q_{r\times n} = \left[f_{ij}^{\mathrm{B}}\right]_{m\times n}$ where $f_{ij}^{\mathrm{B}} = \bigotimes_{k=1}^{r} \mathrm{B}\left(p_{ik}, q_{kj}\right)$.

According to the operators ⊗, A and B used different results can be obtained. To compensate for additive or subtractive noise, operator ⊗ should be replaced either by **max** ($\vee$) or **min** ($\wedge$). We decided to use as operator ⊗ the **median** operator (**med** for short) because as we will demonstrate, it provides excellent results in the presence of mixed noise (additive plus subtractive). If $\mathbf{x}\in \mathbf{Z}^n$ and $\mathbf{y}\in \mathbf{Z}^m$, then $\mathbf{y}\Diamond_{\mathrm{A}}\mathbf{x}^t$ is a matrix of dimensions $m\times n$. Relevant simplifications are obtained when operations $\Diamond_{\mathrm{A}}$ and $\Diamond_{\mathrm{B}}$ are applied between vectors:

1. If $\mathbf{x}\in \mathbf{Z}^n$ and $\mathbf{y}\in \mathbf{Z}^m$, then $\mathbf{y}\Diamond_{\mathrm{A}}\mathbf{x}^t$ is a matrix of dimensions $m\times n$, and also it holds that

$$\mathbf{y}\Diamond_{\mathrm{A}}\mathbf{x}^t = \begin{pmatrix} \mathrm{A}(y_1,x_1) & \mathrm{A}(y_1,x_2) & \cdots & \mathrm{A}(y_1,x_n) \\ \mathrm{A}(y_2,x_1) & \mathrm{A}(y_2,x_2) & \cdots & \mathrm{A}(y_2,x_n) \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{A}(y_m,x_1) & \mathrm{A}(y_m,x_2) & \cdots & \mathrm{A}(y_m,x_n) \end{pmatrix}_{m\times n} .$$

2. If $\mathbf{x}\in \mathbf{Z}^n$ and P a matrix of dimensions $m\times n$, operations $P_{m\times r}\Diamond_{\mathrm{B}}\mathbf{x}$ gives as a result one vector with dimension m, with i-th component $\left(P_{m\times n}\Diamond_{\mathrm{B}}\mathbf{x}\right)_i = \operatorname*{med}_{j=1}^{n} \mathrm{B}\left(p_{ij}, x_j\right)$.

Operators A and B might be chosen among those already proposed in the literature. In this paper we use operators A and B proposed in [6], defined as follows:

$$\mathrm{A}(x,y) = x - y \ . \tag{1.a}$$

$$\mathrm{B}(x,y) = x + y \ . \tag{1.b}$$

3 Kinds of Noises

The proposed memories can cope with several kinds of noises, additive, subtractive and mixed. The case we are interested in, in this paper, is the case of mixed noise. Let $\mathbf{x} \in \mathbf{R}^n$ an input fundamental pattern to an associative memory. Pattern **x** can be altered or corrupted by mixed noise to produce a vector $\tilde{\mathbf{x}}$ by adding or subtracting at random to each component of **x**, x_i a real $c \geq 0$, $\tilde{x}_i = x_i + c$ (additive noise), and $\tilde{x}_i = x_i - c$ (subtractive noise).

4 The Proposal

Two kind of associative memories are proposed, hetero-associative and auto-associative. Due to space limitations, in this paper only auto-associative memories are described. For an auto-associative memory:

1. The fundamental set takes the form $\{(\mathbf{x}^\xi, \mathbf{x}^\xi) \mid \xi = 1,2,\ldots,p\}$.
2. The input and output patterns have the same dimension, for example *n.*
3. The memory is a square matrix.

One auto-associative memory is described. Let us call AS-memory of type **M**.

TRAINING PHASE:

Step 1: For each $\xi = 1,2,\cdots,p$, from each couple $(\mathbf{x}^\xi, \mathbf{x}^\xi)$ build matrix: $\left[\mathbf{x}^\xi \Diamond_{\mathrm{A}} (\mathbf{x}^\xi)^t\right]_{n \times n}$.

Step 2: Apply the median operator to the matrices obtained in Step 1 to get matrix **M** as follows:

$$\mathbf{M} = \operatorname*{\mathbf{med}}_{\xi=1}^{p} \left[\mathbf{x}^\xi \Diamond_{\mathrm{A}} (\mathbf{x}^\xi)^t\right]. \tag{2}$$

The *ij*-th component **M** is given as follows:

$$m_{ij} = \operatorname*{\mathbf{med}}_{\xi=1}^{p} \mathrm{A}(x_i^\xi, x_j^\xi). \tag{3}$$

The two following propositions can be easily demonstrated:

Lemma 1. $\mathbf{M}^\xi, \xi = 1,\ldots p$ is symmetrical.

Lemma 2. **M** is symmetrical.

RECOVERING PHASE:

We have also two cases, i.e.:

Case 1: Recall of a fundamental pattern. A pattern $\mathbf{x}^w$, with $w \in \{1,2,\cdots,p\}$ is presented to the memory **M** and the following operation is done:

$$\mathbf{M}\Diamond_{\mathrm{B}}\mathbf{x}^{w}\ . \tag{4}$$

The result is a column vector of dimension n, with i-th component given as:

$$\left(\mathbf{M}\Diamond_{\mathrm{B}}x^{w}\right)_i = \operatorname*{\mathbf{med}}_{j=1}^{n} \mathrm{B}\left(m_{ij}, x_j^{w}\right)\ . \tag{5}$$

Case 2: Recall of a pattern from an altered version of it. A pattern $\tilde{\mathbf{x}}$ (altered version of a pattern $\mathbf{x}^{w}$ is presented to the auto-associative memory **M** and the following operation is done:

$$\mathbf{M}\Diamond_{\mathrm{B}}\tilde{\mathbf{x}}\ . \tag{6}$$

Again, the result is a column vector of dimension n, with i-th component given as:

$$\left(\mathbf{M}\Diamond_{\mathrm{B}}\tilde{\mathbf{x}}\right)_i = \operatorname*{\mathbf{med}}_{j=1}^{n} \mathrm{B}\left(m_{ij}, \tilde{x}_j\right)\ . \tag{7}$$

The following proposition, not demonstrated here due to space, provides with the conditions for perfect recall of a pattern of the FS:

Theorem 1. Let $\left\{\left(\mathbf{x}^{\alpha},\mathbf{x}^{\alpha}\right) \mid \alpha = 1,2,\ldots,p\right\}$ with $\mathbf{x}^{\alpha} \in \mathbf{R}^{n}$ the fundamental set of an AS-memory **M** and let $\left(\mathbf{x}^{\gamma},\mathbf{x}^{\gamma}\right)$ an arbitrary fundamental couple with $\gamma = 1,\cdots,p$. If $\operatorname*{\mathbf{med}}_{j=1}^{n} \varepsilon_{ij} = 0$, $i = 1,\cdots,m$, $\varepsilon_{ij} = m_{ij} - \mathrm{A}\left(x_i^{\gamma}, x_j^{\gamma}\right)$ then $\left(\mathbf{M}\Diamond_{\mathrm{B}}\mathbf{x}^{\gamma}\right)_i = \mathbf{x}_i^{\gamma}, i = 1\ldots m$.

More restricted conditions are given by the following:

Corollary 1. Let $\left\{\left(\mathbf{x}^{\alpha},\mathbf{x}^{\alpha}\right) \mid \alpha = 1,2,\ldots,p\right\}$, $\mathbf{x}^{\alpha} \in \mathbf{R}^{n}$. An AS-median memory **M** has perfect recall if for all $\alpha = 1,\cdots,p$, $\mathbf{M}^{\alpha} = \mathbf{M}$ where $\mathbf{M} = \mathbf{x}^{\xi}\Diamond_{\mathrm{A}}\left(\mathbf{x}^{\xi}\right)^{t}$ is the associated partial matrix to the fundamental couple $\left(\mathbf{x}^{\alpha},\mathbf{x}^{\alpha}\right)$ and p is the number of couples.

The following proposition, not demonstrated here due to space, provides with the conditions for perfect recall of a pattern of the FS in terns of an altered version of it:

Theorem 2. Let $\left\{\left(\mathbf{x}^{\alpha},\mathbf{x}^{\alpha}\right) \mid \alpha = 1,2,\ldots,p\right\}$, $\mathbf{x}^{\alpha} \in \mathbf{R}^{n}$ a FS with perfect recall. Let $\eta^{\alpha} \in \mathbf{R}^{n}$ a pattern of mixed noise. An AS-memory **M** has perfect recall in the presence of mixed noise if this noise is of median zero, this is if $\operatorname*{\mathbf{med}}_{j=1}^{n} \eta_j^{\alpha} = 0, \forall\alpha$.

5 Numerical Examples

In this section we show how the proposed associative memories are able to recall patterns of the FS and altered versions of them.

Example 1. Recalling the FS. Suppose we want to first memorize and then recall the following FS:

$$\mathbf{x}^1=\begin{pmatrix}0.2\\-0.1\\-0.3\end{pmatrix},\ \mathbf{x}^2=\begin{pmatrix}0.4\\0.1\\-0.1\end{pmatrix}\ \text{and}\ \mathbf{x}^3=\begin{pmatrix}0.6\\0.3\\0.1\end{pmatrix}.$$

TRAINING PHASE:

You can easily verify that:

$$\mathbf{M}=\begin{pmatrix}0.0&0.3&0.5\\-0.3&0.0&0.2\\-0.5&-0.2&0.0\end{pmatrix}.$$

You can also verify that $\forall\alpha,\mathop{\mathbf{med}}_{j=1}^{n}\varepsilon_{ij}=0,\ i=1,\cdots,m$. Perfect recall should be expected. Let us verify this.

RECOVERING PHASE:

$$\mathbf{M}\lozenge_B\mathbf{x}^1=\begin{pmatrix}0.0&0.3&0.5\\-0.3&0.0&0.2\\-0.5&-0.2&0.0\end{pmatrix}\lozenge_B\begin{pmatrix}0.2\\-0.1\\-0.3\end{pmatrix}=\begin{pmatrix}\mathbf{med}[B(0.0,0.2),B(0.3,-0.1),B(0.5,-0.3)]\\\mathbf{med}[B(-0.3,0.2),B(0.0,-0.1),B(0.2,-0.3)]\\\mathbf{med}[B(-0.5,0.2),B(-0.2,-0.1),B(0.0,-0.3)]\end{pmatrix}=\begin{pmatrix}\mathbf{med}(0.2,0.2,0.2)\\\mathbf{med}(-0.1,-0.1,-0.1)\\\mathbf{med}(-0.3,-0.3,-0.3)\end{pmatrix}=\begin{pmatrix}0.2\\-0.1\\-0.3\end{pmatrix}.$$

Similarly:

$$\mathbf{M}\lozenge_B\mathbf{x}^2=\begin{pmatrix}0.0&0.3&0.5\\-0.3&0.0&0.2\\-0.5&-0.2&0.0\end{pmatrix}\lozenge_B\begin{pmatrix}0.4\\0.1\\-0.1\end{pmatrix}=\begin{pmatrix}\mathbf{med}(0.4,0.4,0.4)\\\mathbf{med}(0.1,0.1,0.1)\\\mathbf{med}(-0.1,-0.1,-0.1)\end{pmatrix}=\begin{pmatrix}0.4\\0.1\\-0.1\end{pmatrix}.$$

$$\mathbf{M}\lozenge_B\mathbf{x}^3=\begin{pmatrix}0.0&0.3&0.5\\-0.3&0.0&0.2\\-0.5&-0.2&0.0\end{pmatrix}\lozenge_B\begin{pmatrix}0.6\\0.3\\0.1\end{pmatrix}=\begin{pmatrix}\mathbf{med}(0.6,0.6,0.6)\\\mathbf{med}(0.3,0.3,0.3)\\\mathbf{med}(0.1,0.1,0.1)\end{pmatrix}=\begin{pmatrix}0.6\\0.3\\0.1\end{pmatrix}.$$

Notice how as expected the whole FS has been perfectly recalled.

If a FS satisfies the conditions imposed by Theorem 1, and the level noise added to a pattern $\mathbf{x}^\alpha$ of this FS satisfies Theorem 2, then no matter the level of noise added, the pattern is perfectly recalled. Let us verify this with an example.

Example 2. Suppose we want to recall $\mathbf{x}^2$ from Example 1 given its following distorted version:

$$\tilde{\mathbf{x}}^2=\begin{pmatrix}10.0\\-20.0\\-0.1\end{pmatrix}.$$

Notice the level of noise introduced to components one and two of the pattern, component three was not altered. The median of the noise added to $\mathbf{x}$ equals 0 $\mathbf{med}(9.6,-19.9,0.0)=0.0$. Perfect recall of the pattern should be given:

RECOVERING PHASE:

$$\mathbf{M}\Diamond_B\mathbf{x}^2=\begin{pmatrix}0.0 & 0.3 & 0.5\\ -0.3 & 0.0 & 0.2\\ -0.5 & -0.2 & 0.0\end{pmatrix}\Diamond_B\begin{pmatrix}10.0\\ -20.0\\ -0.1\end{pmatrix}=\begin{pmatrix}\mathbf{med}[B(0.0,10),B(0.3,-20),B(0.5,-0.1)]\\ \mathbf{med}[B(-0.3,10),B(0.0,-20),B(0.2,-0.1)]\\ \mathbf{med}[B(-0.5,10),B(-0.2,20),B(0.0,-0.1)]\end{pmatrix}=\begin{pmatrix}\mathbf{med}(10.0,-19.7,\mathbf{0.4})\\ \mathbf{med}(9.7,-20.0,\mathbf{0.1})\\ \mathbf{med}(9.5,1-9.8,\mathbf{-0.1})\end{pmatrix}=\begin{pmatrix}0.4\\ 0.1\\ -0.1\end{pmatrix}.$$

6 Case of a General FS

6.1 Case of a Non-distorted Pattern

In practice most of the FS of patterns do not satisfy the restricted conditions imposed by Theorem 1 and its Corollary. If this is the case, the patterns of a general FS can be recall by means of the following procedure (an adapted version from the hetero-associative case). Given a general FS not satisfying Theorem 1:

TRAINING PHASE:

Step 1: Transform the FS into an auxiliary fundamental set FS' that satisfies Theorem 1:

```
1) Make $d = const.$
2) Make $\bar{\mathbf{x}}^1=\mathbf{x}^1$, $\hat{\mathbf{x}}^1=0$.
3) For the remaining couples make {
   For $\xi=2$ a $p$ {
      For $i=1$ a $n$ {
            $\bar{\mathbf{x}}_i^\xi=\bar{\mathbf{x}}_i^{\xi-1}+d$ ; $\hat{\mathbf{x}}_i^\xi=\bar{\mathbf{x}}_i^\xi-\mathbf{x}_i^\xi$.
      }
   }
```

Step 2: Build matriz **M** in terms of set FS': Appy to FS' steps 1 and 2 at the begining of Section 2.

RECOVERING PHASE:

We have two cases:

Case 1: Recalling of a patther of the FS:

1) Apply equations (**3**) and (**4**) to each $\bar{\mathbf{x}}^\xi$ of FS' to recall $\bar{\mathbf{x}}^\xi$.
2) Recall each $\mathbf{x}^\xi$ by applying the following inverse transformation: $\mathbf{x}^\xi=\bar{\mathbf{x}}^\xi-\hat{\mathbf{x}}^\xi$.

Case 2: Recalling of a pattern $\mathbf{x}^\xi$ from a distorted version of it, $\tilde{\mathbf{x}}^\xi$:

1) Transform $\tilde{\mathbf{x}}^\xi$ into $\bar{\tilde{\mathbf{x}}}^\xi$ by applying the transformation: $\bar{\tilde{\mathbf{x}}}^\xi=\tilde{\mathbf{x}}^\xi+\hat{\mathbf{x}}^\xi$.
2) Apply equations (**5**) and (**6**) to each $\bar{\tilde{\mathbf{x}}}^\xi$ of FS' to get $\bar{\mathbf{x}}^\xi$.
3) Anti-transform $\bar{\mathbf{x}}^\xi$ as $\mathbf{x}^\xi=\bar{\mathbf{x}}^\xi-\hat{\mathbf{x}}^\xi$ to get $\mathbf{x}^\xi$.

Example 3. Suppose we want to first memorize and then recall the following general fundamental set:

$$\mathbf{x}^1=\begin{pmatrix}0.1\\0.0\\0.2\end{pmatrix},\ \mathbf{x}^2=\begin{pmatrix}0.4\\0.2\\0.5\end{pmatrix}\text{ y }\mathbf{x}^3=\begin{pmatrix}0.6\\0.6\\0.8\end{pmatrix}.$$

TRAINING PHASE:

1) $d=x_1^2-x_1^1=0.4-0.1=0.3$.

2) $\mathbf{x}^1=\begin{pmatrix}0.1 & 0.0 & 0.2\end{pmatrix}'$, $\hat{\mathbf{x}}^1=\begin{pmatrix}0.0 & 0.0 & 0.0\end{pmatrix}'$.

3) $\bar{\mathbf{x}}^2=\begin{pmatrix}0.4\\0.3\\0.5\end{pmatrix}$, $\hat{\mathbf{x}}^2=\begin{pmatrix}0.0\\0.1\\0.0\end{pmatrix}$; $\bar{\mathbf{x}}^3=\begin{pmatrix}0.7\\0.6\\0.8\end{pmatrix}$, $\hat{\mathbf{x}}^3=\begin{pmatrix}0.1\\0.0\\0.0\end{pmatrix}$.

We can easily show that: $\mathbf{M}=\mathbf{M}^1=\mathbf{M}^2=\mathbf{M}^3=\begin{pmatrix}0.0 & 0.1 & -0.1\\-0.1 & 0.0 & -0.2\\0.1 & 0.2 & 0.0\end{pmatrix}$.

RECOVERING PHASE:

Let us consider only case two, which is of more interest. Suppose that we want to recall pattern $\mathbf{x}^2$ from its following distorted version $\tilde{\mathbf{x}}^2=\begin{pmatrix}0.6 & 0.2 & 0.1\end{pmatrix}^T$.

1) As discussed: $\bar{\tilde{\mathbf{x}}}^2=\begin{pmatrix}0.6 & 0.3 & 0.1\end{pmatrix}^T$.

2) $$\mathbf{M}\Diamond_B\bar{\tilde{\mathbf{x}}}^2=\begin{pmatrix}0.0 & 0.1 & -0.1\\-0.1 & 0.0 & -0.2\\0.1 & 0.2 & 0.0\end{pmatrix}\Diamond_B\begin{pmatrix}0.6\\0.3\\0.1\end{pmatrix}=\begin{pmatrix}\mathbf{med}(0.6,0.4,0.0)\\\mathbf{med}(0.5,0.3,0.1)\\\mathbf{med}(0.7,0.5,0.1)\end{pmatrix}=\begin{pmatrix}0.4\\0.3\\0.5\end{pmatrix}=\bar{\mathbf{x}}^2 .$$

Finally, $\mathbf{x}^2=\bar{\mathbf{x}}^2-\hat{\mathbf{x}}^2=\begin{pmatrix}0.4\\0.3\\0.5\end{pmatrix}-\begin{pmatrix}0.0\\0.1\\0.0\end{pmatrix}=\begin{pmatrix}0.4\\0.2\\0.5\end{pmatrix}$.

6.2 Case of a Distorted Pattern

In practice the noise added to a pattern $\mathbf{x}$ of a general FS, does not satisfies the restricted condition imposed by Theorem 2. In this case we propose to use the well-known distance classifier to classify (not to recall) pattern $\mathbf{x}$ from altered version of it. Given an altered version $\tilde{\mathbf{x}}$, of a pattern $\mathbf{x}$ belonging to a general FS, select the index of the class where pattern should be by performing the following operation:

$$\mathbf{x}^j, j = \arg\min_i \left(\min_{k=1}^{n} \left(\left| \tilde{x}_k - x_k^i \right| \right) \right) . \tag{8}$$

Example 4. From example 3, let us take the following altered version of pattern $\mathbf{x}^2$:

$$\tilde{\mathbf{x}}^2 = \begin{pmatrix} 0.8 \\ 0.5 \\ -0.1 \end{pmatrix} .$$

According to the material exposed at the beginning of this section:

For $j = 1$: $\min(|0.8-0.2|, |0.5+0.1|, |-0.1+0.3|) = 0.2$.

For $j = 2$: $\min(|0.8-0.4|, |0.5-0.4|, |-0.1-03|) = 0.1$.

For $j = 3$: $\min(|0.8-0.9|, |0.5-0.7|, |-0.1-0.8|) = 0.2$.

Thus $\tilde{\mathbf{x}}^2$ is put in class number 2.

7 Experiments with Real Patterns

In this section, the proposed auto-associative memories are used to classify real patterns. For this the five voice patterns of the words one to five ("uno" to "cinco" in Spanish) were used. Their size is the same, 599 samples, $x_j^\xi \in \{-0.2, 0.2\}$, $\xi = 1,\ldots,5$, $j = 1,\ldots 1,599$. Let us designate these five images as $\mathbf{x}^1,\ldots\mathbf{x}^5$.

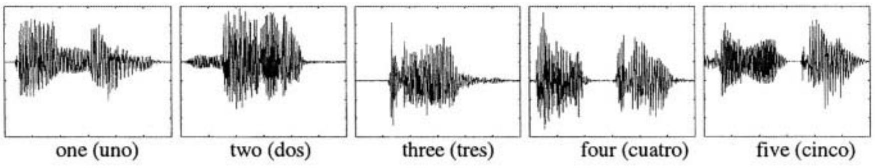


Fig. 1. Images of the five patterns used to test the proposed associative memories

7.1 Construction of the Association Matrix

Each vector, $\mathbf{x}^1,\ldots\mathbf{x}^5$, was then used to construct the corresponding matrices $\mathbf{M}^1,\ldots,\mathbf{M}^5$, by means of the techniques described in Section 4.

7.2 Recalling of the FS

The five patterns were fed to matrix **M** already built. To all of them, the procedure described at the end of section 5 was applied. The five patterns were perfectly recalled.

7.3 Recalling of a Corrupted Pattern

To each vector pattern, $\mathbf{x}^1,\ldots\mathbf{x}^5$ random mixed noise was added to get the distorted vector patterns. To obtain the noisy patterns, to each $x_j^\xi, \xi \in \{1,\ldots,p\}, j \in \{1,\ldots,n\}$ a real number $v, (0.0 \le v \le 0.1)$ was added or subtracted at random. The value $x_j^\xi, \xi \in \{1,\ldots,p\}, j \in \{1,\ldots,n\}$ was changed if $s < t$. $s \in [0,1]$ is an uniformly randomly distributed random variable, t is the parameter controlling how much of the pattern is corrupted. For each pattern, s was varied from 1 to 0.5 with steps of 0.05 to get 55 distorted patterns (11 exemplars for each original pattern). Restriction $\overset{n}{\underset{j=1}{\mathbf{med}}}\eta_j^\alpha = 0$ was not adopted in this case. Fig. 2 shows the last distorted version of each pattern (Notice the level of distortion introduced). These patterns were presented to the trained memory. Equation (8) was applied in this case. In all cases all patterns were correctly recalled regardless of the level noise present.

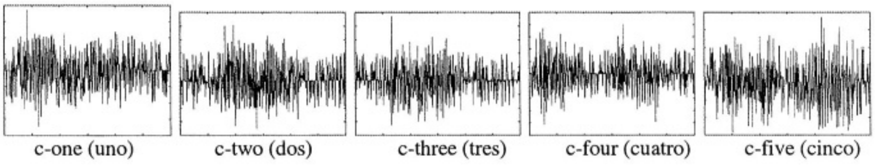


Fig. 2. Last noisy versions (c-xxx) of the patterns used in the experiments

8 Conclusions and Ongoing Research

We described a set of associative memories able to recall patterns altered by mixed noise. The proposed memories are based on the functioning of two operators and the median operator. This is the first time this operator is used to build an associative matrix. This allows us to efficiently tackle with mixed noise. This way we avoid the use of difficult to find so-called kernels.

Numerical and real examples with voice patterns have been provided showing the performance of the proposed memories. Results are very promising. We provided with the conditions under which the proposed memories are able to recall patterns either from the fundamental set of from altered version of them.

Acknowledgements. This work was supported by CIC-IPN, COFAA-IPN, CONACYT under grants 41529 and 40114Y and CGPI-IPN under grant 20020658.

References

[1] K. Steinbuch, Die Lermatrix, *Kybernetik,* 1(1):26-45, 1961.

[2] D. Wilshaw et al. Non-holographic associative memory, Nature 222:960-962, 1969.

[3] J. A. Anderson, A simple neural network generating an interactive memory, *Mathematical Biosciences,* 14:197-220, 1972.

[4] T. Kohonen, Correlation matrix memories, *IEEE Transactions on Computers,* 21(4):353-359, 1972.

[5] J. J. Hopfield, Neural networks and physical systems with emergent collective computational abilities, *Proceedings of the National Academy of Sciences,* 79: 2554-2558, 1982.

[6] G. X. Ritter el al. Morphological associative memories, *IEEE Transactions on Neural Networks,* 9:281-293, 1998.

[7] G. X. Ritter, et al. Morphological bi-directional associative memories, *Neural Networks,* 12:851-867, 1999.

[8] C. Yáñez, Associative Memories based on Order Relations and Binary Operators (In Spanish), PhD Thesis, Center for Computing Research, February of 2002.

[9] P. Sussner. Generalizing operations of binary auto-associative morphological memories using fuzzy set theory. *Journal of Mathematical Imaging and Vision,* 19:81-93, 2003.

[10] G. X. Ritter et al. Reconstruction of patterns from noisy inputs using morphological associative memories. *Journal of Mathematical Imaging and Vision,* 19:95-111, 2003.

[11] H. Sossa, R. Barrón, H. Cortez y Flavio Sánchez. New associative memories for gray-level patterns. 4th International Conference on Control, Virtual Instrumentation and Digital Systems, CICINDI 2002, Pachuca, Mexico. August 26-30, 2002.

Binary Associative Memories Applied to Gray Level Pattern Recalling

Humberto Sossa[1], Ricardo Barrón[1], Francisco Cuevas[2], Carlos Aguilar, and Héctor Cortés[1]

[1] Centro de Investigación en Computación – IPN, Av. Juan de Dios Bátiz s/n, Esquina con Miguel Othón de Mendizábal, Colonia Nueva Industrial Vallejo, C. P. 07700, México, D. F. Mexico
[2] Centro de Investigaciones en Óptica, A. C. Apdo. Postal 1-948, León, Guanajuato, Mexico

Abstract. In this paper we show how a binary memory can be used to recall gray-level patterns. Given a set of gray-level patterns to be first memorized: 1) Decompose each pattern into a set of binary patterns, and 2) Build a binary associative memory (one matrix for each binary layer) with each training pattern set (by layers). A given pattern or a distorted version of it is recalled in three steps: 1) Decomposition of the pattern by layers into its binary patterns, 2) Recovering of each one of its binary components, layer by layer also, and 3) Reconstruction of the pattern from the binary patterns already recalled in step 2. Conditions for perfect recall of a pattern either from the fundamental set or from a distorted version of one them are also given. Experiments are also provided.

1 Introduction

An associative memory **M** is a system that relates vectors, **x** and **y**: $\mathbf{x} \rightarrow \mathbf{M} \rightarrow \mathbf{y}$. Input vector, **x** forms an association with output vector **y**, $(\mathbf{x},\mathbf{y})$. For *k* integer and positive, the corresponding association will be denoted as $\left(\mathbf{x}^k,\mathbf{y}^k\right)$. Associative memory **M** is represented by a matrix whose *ij*-th component is m_{ij}. **M** is generated from a priori finite set of known associations, known as the *fundamental set of associations.* If ξ is an index, the fundamental set is represented as: $\left\{\left(\mathbf{x}^\xi,\mathbf{y}^\xi\right) \mid \xi = 1,2,\ldots,p\right\}$ with *p* the cardinality of the set. The patterns that form the fundamental set are called *fundamental patterns.* If it holds that $\mathbf{x}^\xi = \mathbf{y}^\xi \; \forall \xi \in \{1,2,\ldots,p\}$, **M** is auto-associative, otherwise it is hetero-associative. A distorted version of a pattern **x** will be denoted as $\tilde{\mathbf{x}}$. Several models have been described in the literature, refer for example to [1-6]. In this paper we propose to use the $\alpha\beta$ memories [1], useful in the binary case, to memorize and then recall patterns in the gray level case. Conditions for perfect recalling and examples with real patterns are provided. A first effort in this direction was made in [7].

2 Survey of $\alpha\beta$ Memories

$\alpha\beta$ memories are based on the operation of two operators: α and β, defined as:

$$\alpha : AxA \to B \tag{1}$$

$$\beta : BxA \to A \tag{2}$$

where $A = \{0,1\}$ and $B = \{0,1,2\}$. In tabular form $\alpha : AxA \to B$ and $\beta : BxA \to A$ are defined as shown in Tables 1 and 2:

Table 1. Values of $\alpha(x, y)$

x	y	$\alpha(x, y)$
0	0	1
0	1	0
1	0	2
1	1	1

Table 2. Values of $\beta(x, y)$

x	y	$\beta(x, y)$
0	0	0
0	1	0
1	0	0
1	1	1
2	0	1
2	1	1

Both operations were found by extensive research by taking as foundation the **max** and **min** operations of the morphological associative memories.

2.1 Matrix Operations $\vee_\alpha$, $\wedge_\alpha$, $\vee_\beta$, $\wedge_\beta$

Let $P = \left[p_{ij}\right]_{m\times r}$ and $Q = \left[q_{ij}\right]_{r\times n}$ two matrices. The following matrix operations are defined in [1]:

- α **max** operation: $P_{m\times r} \vee_\alpha Q_{r\times n} = \left[f_{ij}^{\alpha}\right]_{m\times n}$ where $f_{ij}^{\alpha} = \bigvee_{k=1}^{r} \alpha(p_{ik}, q_{kj})$.
- β **max** operation: $P_{m\times r} \vee_\beta Q_{r\times n} = \left[f_{ij}^{\beta}\right]_{m\times n}$ where $f_{ij}^{\beta} = \bigvee_{k=1}^{r} \beta(p_{ik}, q_{kj})$.
- α **min** operation: $P_{m\times r} \wedge_\alpha Q_{r\times n} = \left[f_{ij}^{\alpha}\right]_{m\times n}$ where $h_{ij}^{\alpha} = \bigwedge_{k=1}^{r} \alpha(p_{ik}, q_{kj})$.
- β **min** operation: $P_{m\times r} \wedge_\beta Q_{r\times n} = \left[h_{ij}^{\beta}\right]_{m\times n}$ where $h_{ij}^{\beta} = \bigwedge_{k=1}^{r} \beta(p_{ik}, q_{kj})$.

where $\vee$ and $\wedge$ denote the **max** and **min** operators, respectively. These four matrix operations are similar to the morphological matrix operations described in [6]. When applied between vectors we have:

- If $\mathbf{x} \in A^n$ and $\mathbf{y} \in A^m$, then $\mathbf{y} \vee_\alpha \mathbf{x}^t$ is a matrix of dimensions $m\times n$, and also it holds that

$$\mathbf{y} \vee_\alpha \mathbf{x}^t = \mathbf{y} \wedge_\alpha \mathbf{x}^t = \begin{pmatrix} \alpha(y_1,x_1) & \alpha(y_1,x_2) & \cdots & \alpha(y_1,x_n) \\ \alpha(y_2,x_1) & \alpha(y_2,x_2) & \cdots & \alpha(y_2,x_n) \\ \vdots & \vdots & \ddots & \vdots \\ \alpha(y_m,x_1) & \alpha(y_m,x_2) & \cdots & \alpha(y_m,x_n) \end{pmatrix}_{m\times n} .$$

Symbol $\otimes$ is used to represent both operations, when operating on column vectors: $\mathbf{y} \vee_\alpha \mathbf{x}^t = \mathbf{y} \otimes \mathbf{x}^t = \mathbf{y} \wedge_\alpha \mathbf{x}^t$.

- If $\mathbf{x} \in A^n$ and P a matrix of dimensions $m\times n$, operations $P_{m\times r} \vee_\beta \mathbf{x}$ and $P_{m\times r} \wedge_\beta \mathbf{x}$ give as a result two vectors with dimension m, with i-th component $\left(P_{m\times r} \vee_\beta \mathbf{x}\right)_i = \bigvee_{j=1}^{n} \beta\left(p_{ij}, x_j\right)$ and $\left(P_{m\times r} \wedge_\beta \mathbf{x}\right)_i = \bigwedge_{j=1}^{n} \beta\left(p_{ij}, x_j\right)$.

2.2 $\alpha\beta$ Memories

Two kind of associative memories are described in [1]: hetero-associative and auto-associative. Due to space we only talk about auto-associative memories. If to a hetero-associative is imposed the condition that if $\mathbf{y}^\xi = \mathbf{x}^\xi \ \forall \xi \in \{1,2,\cdots,p\}$ according to Section 1, the memory becomes an auto-associative one. In this case it is obvious that:

- The fundamental set takes the form $\left\{\left(\mathbf{x}^\xi, \mathbf{x}^\xi\right) \mid \xi = 1,2,\ldots,p\right\}$.
- The input and output patterns have the same dimension, for example n.
- The memory is a square matrix.

Two auto-associative $\alpha\beta$ memories: **M** and **W** are fully described in [1]. **M** memories are useful to cope with additive noise; **W** memories, in the contrary, are useful to cope with subtractive noise. Due to space limitations, only **M** memories are described.

Auto-associative $\alpha\beta$ Memories Type M:

TRAINING PHASE:

Step 1: For each $\xi = 1,2,\cdots,p$, from each couple $\left(\mathbf{x}^\xi, \mathbf{x}^\xi\right)$ build the matrix: $\left[\mathbf{x}^\xi \otimes \left(\mathbf{x}^\xi\right)^t\right]_{n\times n}$.

Step 2: Apply the binary **max** operator $\vee$ to the matrices obtained in Step 1 to get matrix **M** as follows:

$$\mathbf{M} = \bigvee_{\xi=1}^{p} \left[\mathbf{x}^\xi \otimes \left(\mathbf{x}^\xi\right)^t\right]. \tag{3}$$

The *ij*-th component **M** is given as follows:

$$m_{ij} = \bigvee_{\xi=1}^{p} \alpha\left(x_i^{\xi}, x_j^{\xi}\right). \tag{4}$$

RECOVERING PHASE: We have two cases:

Case 1: Recovering of a fundamental pattern. A pattern $\mathbf{x}^{\omega}$, with $\omega \in \{1,2,\cdots,p\}$ is presented to the auto-associative memory **M** and the following operation is done:

$$\mathbf{M} \wedge_{\beta} \mathbf{x}^{\omega}. \tag{5}$$

The result is a column vector of dimension *n*, with *i*-th component given as:

$$\left(\mathbf{M} \wedge_{\beta} \mathbf{x}^{\omega}\right)_i = \bigwedge_{j=1}^{n} \beta\left(m_{ij}, x_j^{\omega}\right). \tag{6}$$

In this case the recalling condition is always satisfied due to **M** always have 1's in its main diagonal.

Case 2: Recovering of a pattern from an altered version of it. A pattern $\tilde{\mathbf{x}}$ (altered version with additive noise of a pattern $\mathbf{x}^{\omega}$) is presented to the auto-associative memory **M** and the following operation is done:

$$\mathbf{M} \wedge_{\beta} \tilde{\mathbf{x}}. \tag{7}$$

Again, the result is a column vector of dimension *n*, with *i*-th component given as:

$$\left(\mathbf{M} \wedge_{\beta} \tilde{\mathbf{x}}\right)_i = \bigwedge_{j=1}^{n} \beta\left(m_{ij}, \tilde{x}_j\right). \tag{8}$$

In this case a sufficient condition to obtain $\mathbf{x}^{\omega}$ from $\tilde{\mathbf{x}}$, is that at each row of matrix **M** one of its elements is less or equal to a corresponding element in matrix $\mathbf{x}^{\omega} \otimes \left(\tilde{\mathbf{x}}\right)^t$. For the details, refer to [1].

3 The Proposed Methodology

In this section we show how binary $\alpha\beta$ memories can be used to memorize and recall patterns in more than two gray levels. Without lost of generality, let us just analyze the case of auto-associative memories type **M**.

TRAINING PHASE:

Given *p* monochromatic patterns (possibly images ob objects represented as vectors) $\mathbf{f}_i, i = 1,2,...,p$ with *L* gray levels (for example *L* power of two):

1. Decompose each $\mathbf{f}_i$ in its $n = \log_2 L$ binary patterns $\mathbf{b}_{i,1}, \mathbf{b}_{i,2}, \ldots, \mathbf{b}_{i,n}$, one for each binary plane or layer.

2. Obtain with each set of patterns $\mathbf{b}_{i,j}$, $j = 1,2,\ldots,n$ the corresponding matrix $\mathbf{M}_j$ (training phase), one for each pattern to memorize.

To obtain a given $\mathbf{M}_j$, according to the discussion given in Section 2.2, we have first to obtain the p matrices:

$$\mathbf{b}_{i,j} \otimes (\mathbf{b}_{i,j})^T = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_q \end{pmatrix} \otimes (x_1 \;\; x_2 \;\; \cdots \;\; x_q) = \begin{pmatrix} \alpha(x_1,x_1) & \alpha(x_1,x_2) & \cdots & \alpha(x_1,x_q) \\ \alpha(x_2,x_1) & \alpha(x_2,x_2) & \cdots & \alpha(x_2,x_q) \\ \vdots & \vdots & \ddots & \vdots \\ \alpha(x_q,x_1) & \alpha(x_q,x_2) & \cdots & \alpha(x_q,x_q) \end{pmatrix}. \quad (9)$$

The value of $\alpha(x_s,x_t)$ is obtained by means of Table 1 (Section 2.1). Now, according to the discussion given in Section 2.3, each $\mathbf{M}_j$ is computed as:

$$\mathbf{M}_j = \max_{i=1}^{p} \begin{pmatrix} \alpha(x_1,x_1)_i & \alpha(x_1,x_2)_i & \cdots & \alpha(x_1,x_q)_i \\ \alpha(x_2,x_1)_i & \alpha(x_2,x_2)_i & \cdots & \alpha(x_2,x_q)_i \\ \vdots & \vdots & \ddots & \vdots \\ \alpha(x_q,x_1)_i & \alpha(x_q,x_2)_i & \cdots & \alpha(x_q,x_q)_i \end{pmatrix}. \quad (10)$$

RECOVERING PHASE:

For easy of explanation we will treat with the case of recovering a fundamental pattern. Given a pattern $\mathbf{f}_k$ to recover. Its belongs to the set of fundamental patterns:

1. Decompose $\mathbf{f}_k$, in its n binary patterns $\mathbf{b}_{k,1}, \mathbf{b}_{k,2}, \ldots, \mathbf{b}_{k,n}$, one for each plane.
2. Recall each $\mathbf{b}_{k,j}$, through each $\mathbf{M}_j$. According to the discussion given in Section 2.3, the component $\mathbf{b}_{k,j}$ is recovered by means of the following product:

$$\mathbf{M}_j \wedge_\beta \mathbf{b}_{i,j} = \begin{pmatrix} m_{1,1} & m_{1,2} & \cdots & m_{1,q} \\ m_{2,1} & m_{2,2} & \cdots & m_{2,q} \\ \vdots & \vdots & \ddots & \vdots \\ m_{q,1} & m_{q,2} & \cdots & m_{q,q} \end{pmatrix} \wedge_\beta \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_q \end{pmatrix} = \begin{pmatrix} \beta(m_{1,1},x_1) \wedge \beta(m_{1,2},x_2) \wedge \cdots \wedge \beta(m_{1,q},x_q) \\ \beta(m_{2,1},x_1) \wedge \beta(v_{2,2},x_2) \wedge \cdots \wedge \beta(v_{2,q},x_q) \\ \vdots \\ \beta(m_{q,1},x_1) \wedge \beta(m_{q,2},x_2) \wedge \cdots \wedge \beta(m_{q,q},x_q) \end{pmatrix} \quad (11)$$

where the value of $\beta(m_{r,s},x_t)$ ($\beta(w_{r,s},x_t)$) is obtained from Table 2 (Section 2.1).

3. Recover the pattern through the "sum" of the recalled partial patterns, in this case the operation opposite to decomposition. Pattern $\mathbf{f}_k$ is recovered in terms of its n binary components, $\mathbf{b}_{k,i}$ as:

$$\mathbf{f}_k = \sum_{i=1}^{n} 2^i \cdot \mathbf{b}_{k,i} \quad (12)$$

The proposed technique is supported on the one-to-one relation between integer numbers describing the patterns and their binary representation. The following proposition provides the conditions under which the proposed can be used to perfectly recall patterns either from the fundamental set or from altered versions of them.

Theorem 3.1. *Let* $\{(\mathbf{y}^\xi,\mathbf{x}^\xi) \mid \xi = 1,2,\ldots,p\}, \mathbf{x}^\xi, \mathbf{y}^\xi \in \mathbf{Z}_L^m$ *a fundamental set of patterns in L gray levels, then the set presents perfect recall and with noise if this happens at each plane of bits of the binary decomposition of the patterns. The binary decomposition of a pattern* $\mathbf{x}^\xi$ *in vector form is given as* $\mathbf{x}^\xi = \sum_{k=0}^{n-1} 2^k x^{\xi_k}, L = 2^n$ *with* $\mathbf{x}^{\xi_k} \in \{0,1\}^m$*; in this case we say that* $\mathbf{x}^{\xi_k}$ *belongs to the k-th plane of bits of the binary decomposition.*

4 Numerical Results

Example 4.1. Let us suppose that $L=4$, consider the following three patterns to be memorized and the recovered by means of an auto-associative memory type **M**:

$$\mathbf{f}_1 = \begin{pmatrix}1\\2\\3\\1\end{pmatrix}, \mathbf{f}_2 = \begin{pmatrix}1\\0\\2\\1\end{pmatrix} \text{ and } \mathbf{f}_3 = \begin{pmatrix}2\\2\\2\\2\end{pmatrix}.$$

According to the material exposed in Section 3:

TRAINING PHASE:

Decomposition of each pattern into its different layers:

Because $L = 4$ then $n = 2$. Then:

$$\mathbf{b}_{1,1} = \begin{pmatrix}1\\0\\1\\1\end{pmatrix}, \mathbf{b}_{1,2} = \begin{pmatrix}0\\1\\1\\0\end{pmatrix}, \mathbf{b}_{2,1} = \begin{pmatrix}1\\0\\0\\1\end{pmatrix}, \mathbf{b}_{2,2} = \begin{pmatrix}0\\0\\1\\0\end{pmatrix}, \mathbf{b}_{3,1} = \begin{pmatrix}0\\0\\0\\0\end{pmatrix} \text{ and } \mathbf{b}_{3,2} = \begin{pmatrix}1\\1\\1\\1\end{pmatrix}.$$

Obtaining of the corresponding matrices $\mathbf{M}_1$ and $\mathbf{M}_2$, one for each layer. According to material exposed presented in Section 3:

$$\mathbf{b}_{1,1} \otimes (\mathbf{b}_{1,1})^T = \begin{pmatrix}1\\0\\1\\1\end{pmatrix} \otimes (1\ 0\ 1\ 1) = \begin{pmatrix}\alpha(1,1) & \alpha(1,0) & \alpha(1,1) & \alpha(1,1)\\ \alpha(0,1) & \alpha(0,0) & \alpha(0,1) & \alpha(0,1)\\ \alpha(1,1) & \alpha(1,0) & \alpha(1,1) & \alpha(1,1)\\ \alpha(1,1) & \alpha(1,0) & \alpha(1,1) & \alpha(1,1)\end{pmatrix} = \begin{pmatrix}1 & 2 & 1 & 1\\ 0 & 1 & 0 & 0\\ 1 & 2 & 1 & 1\\ 1 & 2 & 1 & 1\end{pmatrix},$$

You can also easily show that:

$$\mathbf{b}_{2,1}\otimes(\mathbf{b}_{2,1})^T=\begin{pmatrix}1&2&2&1\\0&1&1&0\\0&1&1&0\\1&2&2&1\end{pmatrix},\ \mathbf{b}_{3,1}\otimes(\mathbf{b}_{3,1})^T=\begin{pmatrix}1&1&1&1\\1&1&1&1\\1&1&1&1\\1&1&1&1\end{pmatrix},\ \mathbf{b}_{1,2}\otimes(\mathbf{b}_{1,2})^T=\begin{pmatrix}1&0&0&1\\2&1&1&2\\2&1&1&2\\1&0&0&1\end{pmatrix},$$

$$\mathbf{b}_{2,2}\otimes(\mathbf{b}_{2,2})^T=\begin{pmatrix}1&1&0&1\\1&1&0&1\\2&2&1&2\\1&1&0&1\end{pmatrix},\text{ and }\mathbf{b}_{3,2}\otimes(\mathbf{b}_{3,2})^T=\begin{pmatrix}1&1&1&1\\1&1&1&1\\1&1&1&1\\1&1&1&1\end{pmatrix}.$$

Now, according to Equation (10):

$$\mathbf{M}_1=\begin{pmatrix}1&2&1&1\\0&1&0&0\\1&2&1&1\\1&2&1&1\end{pmatrix}\vee\begin{pmatrix}1&2&2&1\\0&1&1&0\\0&1&1&0\\1&2&2&1\end{pmatrix}\vee\begin{pmatrix}1&1&1&1\\1&1&1&1\\1&1&1&1\\1&1&1&1\end{pmatrix}=\begin{pmatrix}1&2&2&1\\1&1&1&1\\1&2&1&1\\1&2&2&1\end{pmatrix}\text{ and }\mathbf{M}_2=\begin{pmatrix}1&0&0&1\\2&1&1&2\\2&1&1&2\\1&0&0&1\end{pmatrix}\vee\begin{pmatrix}1&1&0&1\\1&1&0&1\\2&2&1&2\\1&1&0&1\end{pmatrix}\vee\begin{pmatrix}1&1&1&1\\1&1&1&1\\1&1&1&1\\1&1&1&1\end{pmatrix}=\begin{pmatrix}1&1&1&1\\2&1&1&2\\2&2&1&2\\1&1&1&1\end{pmatrix}.$$

RECOVERING PHASE:

Recovering, for example, of pattern: $\mathbf{f}_1=\begin{pmatrix}1\\2\\3\\1\end{pmatrix}$. Knowing that L=4:

Decomposition of the pattern into its different layers: $\mathbf{b}_{1,1}=\begin{pmatrix}1\\0\\1\\1\end{pmatrix}$ and $\mathbf{b}_{1,2}=\begin{pmatrix}0\\1\\1\\0\end{pmatrix}$.

Recovering of the pattern at each level. According to Equation (11):

$$\mathbf{M}_1\wedge_\beta\mathbf{b}_{1,1}=\begin{pmatrix}1&2&2&1\\1&1&1&1\\1&2&1&1\\1&2&2&1\end{pmatrix}\wedge_\beta\begin{pmatrix}1\\0\\1\\1\end{pmatrix}=\begin{pmatrix}1\\0\\1\\1\end{pmatrix}\text{ and }\mathbf{M}_2\wedge_\beta b_{1,2}=\begin{pmatrix}1&1&1&1\\2&1&1&2\\2&2&1&2\\1&1&1&1\end{pmatrix}\wedge_\beta\begin{pmatrix}0\\1\\1\\0\end{pmatrix}=\begin{pmatrix}0\\1\\1\\0\end{pmatrix}.$$

Final reconstruction of the total pattern by using Equation (12):

$$\mathbf{f}_1=\begin{pmatrix}2^0\times1+2^1\times0\\2^0\times0+2^1\times1\\2^0\times1+2^1\times1\\2^0\times1+2^1\times0\end{pmatrix}=\begin{pmatrix}1\\2\\3\\1\end{pmatrix}.$$

As you can see the fundamental pattern is perfectly recovered. This as we saw is one of the properties of the $\alpha\beta$ associative memories. The reader can easily verify that the other two patterns are also perfectly recovered.

Example 4.2. Recalling of a fundamental pattern from a noisy version of it using an auto-associative memory of type **M**.

According to material discussed in Section 2, **M** memories work well with additive noise (not with any additive noise) but with additive noise such as all components at the different layers of the decomposed pattern change from 0 to 1, but not inversely. If this does not happen then perfect recall could not occur. Let us consider the following distorted version of pattern $\mathbf{f}_1$ and its decomposition:

$$\tilde{\mathbf{f}}_1 = \begin{pmatrix} 3 \\ 2 \\ 3 \\ 1 \end{pmatrix} \text{ and } \tilde{\mathbf{b}}_{1,1} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix} \text{ and } \tilde{\mathbf{b}}_{1,2} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \end{pmatrix}.$$

Notice how in this case at both layers, and for the same pixel, additive noise has been also added to the corresponding bits. In particular bit number 1 of pattern $\tilde{\mathbf{b}}_{1,2}$ has changed from 0 to 1. Bit number 1 of pattern $\tilde{\mathbf{b}}_{1,1}$ has no modifications. Because additive noise has been also added to each binary layer of the decomposition, we should expect perfect recall according to Theorem 3.1. Let us verify this.

Recovering of the pattern at each level. According to Equation (12):

$$\mathbf{M}_1 \wedge_\beta \tilde{\mathbf{b}}_{1,1} = \begin{pmatrix} 1 & 2 & 2 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 2 & 1 & 1 \\ 1 & 2 & 2 & 1 \end{pmatrix} \wedge_\beta \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix} \text{ and } \mathbf{M}_2 \wedge_\beta \tilde{\mathbf{b}}_{1,2} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & 1 & 2 \\ 2 & 2 & 1 & 2 \\ 1 & 1 & 1 & 1 \end{pmatrix} \wedge_\beta \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}.$$

Notice how the fundamental binary patterns were perfectly recovering. From this we can wait that the recalling of the total pattern will be also possible, let us see.

Final reconstruction of the pattern (Equation (12)):

$$\tilde{\mathbf{f}}_1 = \begin{pmatrix} 2^0 \times 1 + 2^1 \times 0 \\ 2^0 \times 0 + 2^1 \times 1 \\ 2^0 \times 1 + 2^1 \times 1 \\ 2^0 \times 1 + 2^1 \times 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 1 \end{pmatrix} = \mathbf{f}_1.$$

From this example we can see that the recovering of a pattern with $L > 2$ (according to Theorem 3.1) depends on recovering of its binary patterns.

5 Experiments with Real Patterns

In this section the proposed extension is tested with more realistic patterns. Images of five real objects (a bolt, a washer, an eyebolt, a hook and a dovetail) were used are shown in Figure 1. The images are 32 by 29 pixels and 256 gray levels. Only $\alpha\beta$ auto-associative memories of type **M** were tested.

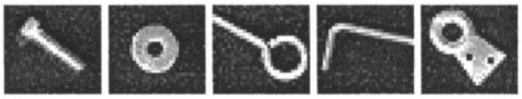


Fig. 1. Images of the five objects used in to test the proposed extension

5.1 Construction of the Association Matrices

The five images were first converted to vectors of 968 elements (32 times 29) each one. These vectors were next decomposed into their 8 binary layers. They were finally used to construct the 8 matrices $\mathbf{M}_1, \mathbf{M}_2, \ldots, \mathbf{M}_8$, by using the corresponding equations given in Section 3.

5.2 Recovering of a Pattern by a Corrupted Version of It

Three groups of images were generated: The first one with additive noise, the second one with saturated noise of type salt, and the third one with manually added saturated noise. In the first case, to the gray-value $f(x, y)$ of pixel with coordinates (x, y), an integer v was added, such that $f(x,y)+v \leq (L-1)$. In the second case, again to the gray-value of a pixel an integer v was added, such that $f(x, y)+v = (L - 1)$. The value $f(x, y)$ of the gray-level of the pixel was

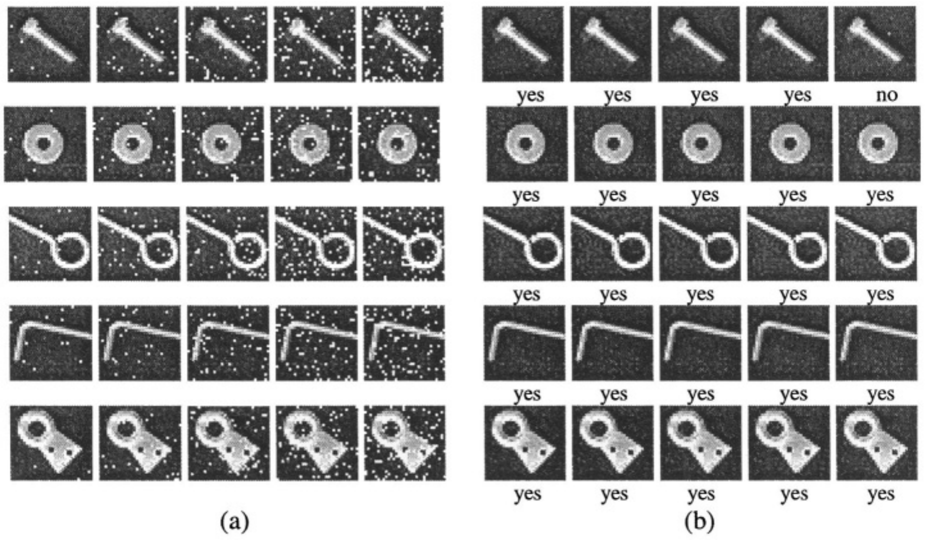


Fig. 2. (a) Versions with salt noise. (b) Recalled versions

changed if $s < t$. $s \in [0,1]$ is an uniformly randomly distributed random variable, t is the parameter controlling how much of the image is corrupted. In the third case Microsoft PAINT utility was used. Only second experiment is shown.

Case with Salt Noise. Twenty five images were obtained as explained. Parameter s was varied form 0.99 to 0.91 in steps of 0.02. The value of the pixel was saturated to $L-1$. Figure 3(a) shows the obtained images. In this case, in some cases the desired image was correctly recalled, in other cases it was not. Recalled versions are shown in Figure 3(b). Under each recalled image it is indicated if it matches the original image or not. Notice how despite the level of noise introduced is bigger than in the first case, recalled versions match better with the originals. This is normal due to saturation noise assures that the values of the components of the patterns change form 0 a 1.

It is worth to mention that the average time to recall an image in all cases is of 6 seconds in a Pentium 4 running at 1.3 GHz.

6 Conclusion and Ongoing Research

We have shown how is possible to use binary memories, to efficiently recall patterns in more than two gray levels. This is possible due to a pattern can be decomposed into binary patterns. In this case the $\alpha\beta$ memories are applied by layers, combining the partial results for the recovering of a pattern on posterior steps. One feature of the proposed extension is that layers can be processed in parallel allowing huge computation time reductions. It is worth to mention that the proposed technique can be adapted to any kind of binary associative memories while their input patterns can be decomposed into binary layers.

Actually, we are investigating how to use the proposed approach in the presence of mixed noise and other variations. We are also working through an application that allows us to operate with bigger images and other patterns such a voice where the proposal would have a more realistic application. We are also working toward the proposal of new associative memories able to operate with gray level patterns where no binary layer decomposition is needed.

Acknowledgements. This work was supported by CIC-IPN, COFAA-IPN, CONACYT under grants 41529 and 40114Y and CGPI-IPN under grant 20020658.

References

[1] C. Yáñez, Associative Memories based on Order Relations and Binary Operators (In Spanish), PhD Thesis, Center for Computing Research, February of 2002.

[2] K. Steinbuch, Die Lermatrix, Kybernetik, 1(1):26-45, 1961.

[3] J. A. Anderson, A simple neural network generating an interactive memory, Mathematical Biosciences, 14:197-220, 1972.

[4] T. Kohonen, Correlation matrix memories, IEEE Transactions on Computers, 21(4):353-359, 1972.

[5] J. J. Hopfield, Neural networks and physical systems with emergent collective computational abilities, Proceedings of the National Academy of Sciences, 79: 2554-2558, 1982.

[6] G. X. Ritter, P. Sussner and J. L. Díaz de León, Morphological associative memories, IEEE Transactions on Neural Networks, 9:281-293, 1998.

[7] R. Barrón, H. Sossa and H. Cortés, Associative processing of patterns in gray levels using binary decomposition and α-β memories. XI CIC 2002, Mexico, City, November 25 to 29, 2002, pp. 415-426.

Selection of an Automated Morphological Gradient Threshold for Image Segmentation. Application to Vision-Based Path Planning*

F.A. Pujol[1], P. Suau[2], M. Pujol[2], R. Rizo[2], and M.J. Pujol[3]

[1]Depto. Tecnología Informática y Computación
fpujol@dtic.ua.es
[2]Depto. Ciencia de la Computación e Inteligencia Artificial
{pablo, mar, rizo}@dccia.ua.es
[3]Depto. Matemática Aplicada,
Universidad de Alicante, Ap. Correos 99, 03080 Alicante, España
mjose@ua.es

Abstract. Segmentation is an essential part of practically any automated image recognition system, since it is necessary for further processing such as feature extraction or object recognition. There exist a variety of techniques for threshold selection, as it is a fast and robust method. Threshold value will have considerable effects on the boundary position and overall size of the extracted objects. In this work, we propose an automated thresholding selection, which considers the local properties of a pixel. To do this, the algorithm calculates the morphological gradient and Laplacian and, afterwards, chooses a suitable threshold after estimating the lowest distance between the ideal segmentation and the morphological gradient thresholding segmentation. As an application of this segmentation process, we have implemented a path planning algorithm for a vision-based system. The experiments show that our path planning algorithm is able to find good solution paths after a training process.

1 Introduction

In recent years, an increasing amount of robotic research has focused on the problem of planning and executing motion tasks autonomously, i.e., without human guidance; e.g., see [1], [2]. Many different path planning methods have been proposed over the last few years, although there are only a few that deal with computer vision techniques.

In relation to this, the recognition of objects in images is a non-trivial task. Machine learning research involves training a classifier with data obtained from known images, and then predicting the label of test samples from unseen images.

* This work has been supported by the Spanish *Comisión Interministerial de Ciencia y Tecnología* (CICYT), project number TIC2001-0245-C02-02 and by the Generalitat Valenciana, projects numbers GV04B685,GV04B634.

The task of decomposing an image into distinctly different but homogeneous parts is called image segmentation, and the process of extracting information from these regions that define their unique properties is called feature extraction.

Image segmentation has been the subject of intensive research and a wide variety of segmentation techniques have been reported in the literature (see [3]). It is a crucial issue since it is the first step of the image understanding process, and all others steps, such as feature extraction and recognition, depend heavily on its results.

In general, the segmentation algorithms are based on two important criteria: the homogeneity of the region and the discontinuity between adjacent disjoint regions. Although many image segmentation approaches deal with these problems ([4], [5]), it is difficult to satisfy all the properties for the optimal set of segmented regions. The resulting segmented images generally depend on the predetermined threshold values. So the algorithm often fails due to it either merges regions that must be divided or separates connected regions.

Consequently, in this work, we propose a fast, robust method for image segmentation that takes into account some local properties of every pixel to calculate a well-suited threshold; thus, our segmentation scheme gives good results even when the tests are performed with noisy real-world images. As a result, this paper shows a method to implement efficiently some path planning tasks for a robot using our segmentation technique, where the environmental features play a major role for completing the task.

The paper is organized as follows: Firstly, we define a method to obtain the morphological gradient threshold (MGT) image segmentation in Section 2. As a result, each of the steps necessary to develop this algorithm are explained in Section 3. Then, Section 4 establishes a measure related to the appropriateness of our technique. Afterwards, in Section 5 the experimentation is widely illustrated to verify the accomplishment of our research objectives. As an application of our segmentation algorithm, a method to develop collision-free paths using is described in Section 6. Finally, we conclude with some important remarks in Section 7.

2 MGT Algorithm for Image Segmentation

Designing an image segmentation scheme needs the consideration of two features of visual recognition: cost and uncertainty. Visual recognition is generally costly because the image data is large. Visual information contains uncertainty from many sources such as discretization. In general, the more observation a vision system performs, the more information is obtained, but the more cost is required. Thus, a trade-off must be considered between the cost of visual recognition and the effect of information to be obtained by recognition. In relation to this, some of the most popular approaches which provide low computation times and good information are the threshold techniques and the edge-based methods [6].

Threshold techniques, which make decisions based on local pixel information, are effective when the intensity levels of the objects are exactly outside the range of the levels in the background. These thresholding algorithms are simple and give very good results, but deciding the threshold values is not easy. Specially, this is a really serious problem for an automated vision system, as the system should decide the threshold values taking its own decision.

On the other hand, edge-based methods (e.g., gradient operators) focus on contour detection. They involve finding the edges of objects in the image and using this edge information to achieve the complete boundaries for the main objects in the image. Edge detection has many problems, especially when working with noisy images, since it could even fragment the true edges.

To overcome these problems, we propose a method that combines both thresholding and gradient operators: the so-called Morphological Gradient Threshold (MGT) segmentation, as described in Table 1. It consists of 7 main steps, where the gradient and the Laplacian are calculated in terms of Mathematical Morphology operations and the optimal threshold value is selected by measuring the lowest distance between the ideal segmentation and a collection of MGT segmented images.

Table 1. MGT Segmentation Algorithm

```
Step 1. Image smoothing.
Step 2. Global dilation and erosion.
Step 3. For every pixel, create a list of symbols by means of the Morphological
        gradient and the Morphological Laplacian.
Step 4. Creation of a pixel-symbol map.
Step 5. Binarization of the pixel-symbol map.
Step 6. Computation of a suitable measure to obtain the optimal threshold.
Step 7. Obtention of the MGT segmented image.
```

Next, a more detailed description of this process is found.

3 Construction of a Pixel-Symbol Map

3.1 Image Smoothing and Derivative-Based Operations

In every digital image there is a certain amount of white noise. To avoid the noise effects, which only consume computation time and affect the real image features, an initial filtering process has to be applied. There are many algorithms to accomplish this task; in our approach a Gaussian filter has been chosen, since it preserves many of the image features while its computational cost can be assumed in a real-time environment. For more information see [7].

Once the noise is eliminated, the point is how to create the pixel-symbol map. To do this, let us consider first the computation of some derivative-based operations, i.e., the gradient and the Laplacian.

As mentioned before, edge detection is a main problem in image analysis. There are many approaches to obtain edges by means of the gradient of an image (e.g., Prewitt or Sobel operators). Among all of these methods we find the morphological gradient, which uses the Mathematical Morphology operators.

Mathematical Morphology (MM) is a geometric approach in image processing and, originally, it was developed as a tool for shape analysis. The two elementary operations in MM are erosion and dilation [8]; from this two primitives, more complex operations are constructed, such as opening, closing and top-hat. Therefore, one can define the morphological gradient of an image *X* by a structuring element (SE) *B*, $\rho_B(X)$, as:

$$\rho_B(X) = \frac{1}{2}(\delta_B(X) - \varepsilon_B(X)) . \tag{1}$$

where $\delta_B(X)$ and $\varepsilon_B(X)$ are, respectively, the dilation and the erosion of an image *X* by a SE *B*.

It is well-known that MM has not been usually utilized when working with real-time images, as it has a very high running time. In [8] it is described an algorithm which overcome this problem by taking into account all the pixels in an image at the same time.

Let us consider that an image is dilated, using a line-segment SE that consists of two points separated by n pixels. Then, we define a projection matrix, M_p, built from the original image moved $n + 1$ points to the right/left, depending on the location of the SE center. Considering a binary image, one should do the logical operation OR between the original image and the moved one in order to obtain the required dilation. This method can be extended to dilations with any 2-D SE and, more generally, to the erosion, where the logical operation AND is used instead of the OR one. In gray-scale images, OR/AND operations should be replaced by the supremum/infimum ones. The computational cost of this approach is even 50% better than the classical MM method [8].

The following step is to calculate the second derivative, the Laplacian. Again, we have chosen a morphological implementation for the Laplacian, as we can use with costless time the previously pre-calculated erosion and dilation. Thus, the morphological Laplacian of an image *X* by a SE *B*, $\Lambda_B(X)$, is defined as:

$$\Lambda_B(X) = \frac{1}{2}(\delta_B(X) + \varepsilon_B(X)) - X . \tag{2}$$

The results for a gray-scale image after these initial steps are shown in Fig. 1, where the SE *B* is a 3×3 square.

3.2 The Pixel-Symbol Map

The next task is building a map that characterizes properly the pixels for a good segmentation. Thus, the pixel-symbol map $m(x, y)$ is obtained as follows:

$$m(x,y) = \begin{cases} 128 & \text{if} \quad \rho_B(x,y) < MGT, \\ 255 & \text{if} \quad \rho_B(x,y) \geq MGT \text{ and } \Lambda_B(x,y) \geq 0, \\ 0 & \text{if} \quad \rho_B(x,y) \geq MGT \text{ and } \Lambda_B(x,y) < 0. \end{cases} \tag{3}$$

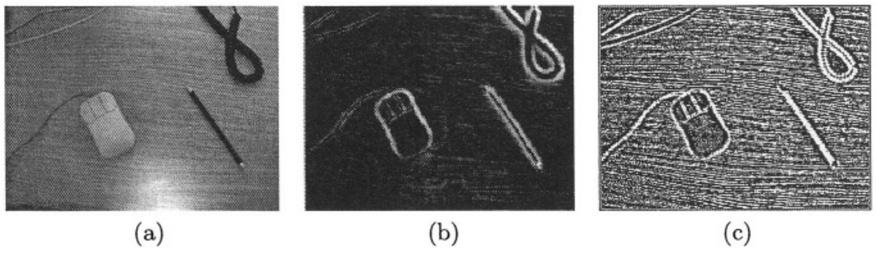


Fig. 1. A real image: (a) Original image. (b) $\rho_B(X)$. (c) $\Lambda_B(X)$

where *MGT* is the morphological gradient threshold and (x, y) is a pixel in *X*. The resulting image has three different gray-levels, according to if a pixel belongs to an object, to the background or to the borders.

The choice of the threshold value is one of the most difficult tasks, since the final result is high dependent on many factors, such as lighting conditions, objects texture or shading. Fig. 2 shows the results of the construction of the pixel-symbol map for the image in Fig. 1, with several different *MGT* values.

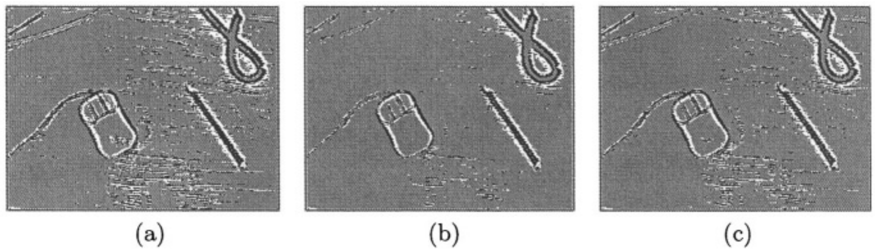


Fig. 2. The pixel-symbol map $m(x, y)$ with different *MGT* values: (a) Gradient mean. (b) $MGT = 0.9 * \max(\rho_B(x, y))$. (c) $MGT = 0.8 * \max(\rho_B(x, y))$

Though many practical systems utilize an experimentally obtained threshold, in this work we consider the use of an automated thresholding system. This method takes into account a binary image metrics to compare the segmentation results and, afterwards, to establish the quality level of the obtained segmentation, as it is described in the following section.

4 A Measure of the Quality of the Segmentation

A main problem in computer vision is to be able to compare the results using a proper metrics. This will quantify the differences between two images and, if binary images are used, the method would be both easily implementable and low computationally complex. In our system we are interested in measuring the distance between image *G* (the map after gradient thresholding) and image *A* (the ideal segmentation). Thus, it will establish the optimal *MGT* value.

Hence, the map must be binarized first. To do this, we must recall that $m(x,y)$ has only 3 gray-levels (Eq. (3)): 0, 128 and 255. For simplicity, let us consider that the threshold is the same as in the construction of $m(x,y)$, i.e., the gradient threshold *MGT*. The results of this process are shown in Fig. 3.

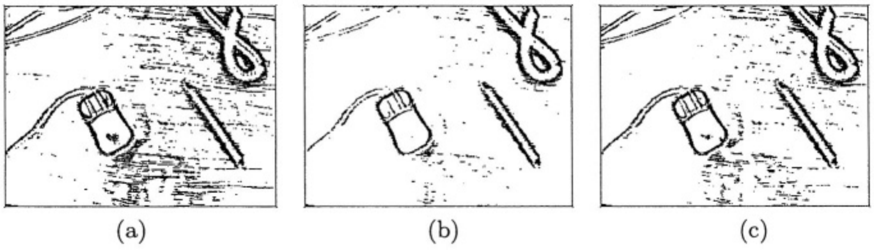


Fig. 3. Binarization with different *MGT* values: (a) Gradient mean. (b) $MGT = 0.9 * \max(\rho_B(x,y))$. (c) $MGT = 0.8 * \max(\rho_B(x,y))$

Next, a reliable measure to compare the obtained image segmentation with an ideal segmentation must be selected.

As proved in [9], a good error measurement for binary images is $\Delta^p(A,G)$, defined as the p^{th} order mean difference between the thresholded distance transforms of two images: A (the ideal segmentation) and G (the binary pixel-symbol map). Let us define first some previous terms:

- Let X denote the pixel raster.
- A binary image $A \subseteq X$ is a set $A = \{x \in X : A(x) = 1\}$.

If $\varrho(x,y)$ is the distance between two pixels x and y, the shortest distance between a pixel $x \in X$ and $A \subseteq X$ is defined as:

$$d(x,A) = \inf\{\varrho(x,a) : a \in A\} \ . \tag{4}$$

Then, for $1 \leq \varrho \leq \infty$ we define:

$$\Delta^p(A,G) = \left[\frac{1}{N}\sum_{x \in X} |w(d(x,A)) - w(d(x,G))|^p\right]^{1/p} \ . \tag{5}$$

where N is the total number of pixels in X and $w(t) = \min(t,c)$, for $c > 0$.

Intuitively, $\Delta^p(A,G)$ measures the suitability of an estimated image to be used instead of the real one.

Now, we can evaluate the goodness of our segmentation scheme.

5 Experiments

Let us show now the results of some experiments completed for our model. The tests have been performed with a set of real images, whose pixel-symbol maps have been calculated for different *MGT* values. Then, after applying the binarization process, the distance $\Delta^p(A,G)$ has been computed.

Table 2 shows the results for the image in Fig. 1, where $p = 2, c = 5$.

Table 2. Results Obtained After the Segmentation Process

MGT value	Distance $\Delta^p(A,G)$
Gradient mean	0.6038
$MGT = 0.95 * \max(\rho_B(x,y))$	0.2722
$MGT = 0.9 * \max(\rho_B(x,y))$	0.1988
$MGT = 0.85 * \max(\rho_B(x,y))$	0.3412
$MGT = 0.8 * \max(\rho_B(x,y))$	0.3704

As shown, the lowest distance is obtained when $MGT = 0.9 * \max(\rho_B(x,y))$. Fig. 4 compares the ideal segmentation and the *MGT* segmentation with the lowest $\Delta^p(A,G)$. Intuitively, if we compare the previous results in Fig. 3, the selected *MGT* value is quite similar to the ideal segmentation.

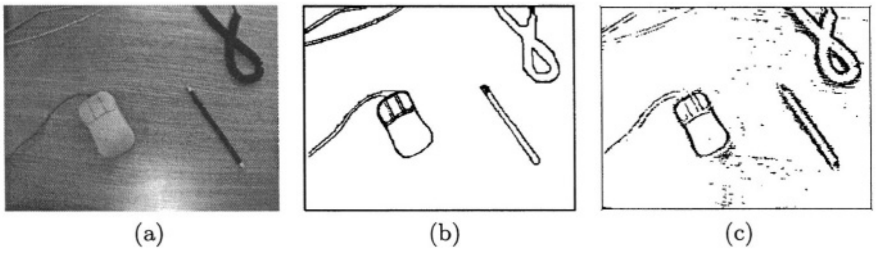


Fig. 4. (a) Original image. (b) Ideal segmentation. (c) *MGT* segmentation

Let us consider now a more complex real image in order to confirm the accuracy of our technique to give an automated extraction of the threshold value with the best behavior. Fig. 5 and Table 3 show the results.

The minimum distance is obtained again when $MGT = 0.9 * \max(\rho_B(x,y))$ and, as a consequence, we can conclude that the parameters used for this segmentation are near optimal, as they have a behavior very close to ideal segmentation.

Nevertheless, the threshold could be adaptively updated so as to assume the real conditions in which every image has been taken by the vision system.

6 A Vision-Based Path Planning Algorithm

Designing a path planning algorithm for a vision-guided robot needs the consideration of two features of visual recognition: cost and uncertainty. In general,

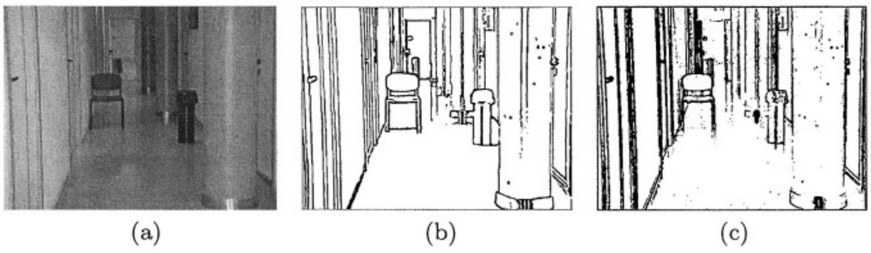


Fig. 5. (a) Original image. (b) Ideal segmentation. (c) *MGT* segmentation

the more observation a robot performs, the more information is obtained, but the more cost is required. Thus, a trade-off must be considered between the cost of visual recognition and the effect of information to be obtained by recognition.

Table 3. Results Obtained After the Segmentation Process

MGT value	Distance $\Delta^p(A, G)$
Gradient mean	0.5526
$MGT = 0.95 * \max(\rho_B(x,y))$	0.3115
$MGT = 0.9 * \max(\rho_B(x,y))$	0.2245
$MGT = 0.85 * \max(\rho_B(x,y))$	0.2731
$MGT = 0.8 * \max(\rho_B(x,y))$	0.3219

Consequently, in this section we propose a vision-based path planning algorithm, taking into account the segmentation scheme discussed before. This process can be described as:

1. Apply the *MGT* segmentation algorithm to the initial image. As a result, obstacle pixels are labelled as '0' and free-space pixels as '1'.
2. Repeat the following steps:
   - Use a morphological dilation.
   - Convert black tones into a pre-determined grey tone, whose intensity is increased for each iteration.
3. Obtain a path map where higher intensity pixels are the obstacle-free zones.

To accomplish the task, from the starting point of the path, our algorithm chooses the pixel with the lowest probability of collision in a 3 × 3 neighborhood. To do this, we define a set of weights w_i that are estimated as:

$$w_i = \exp\{a * (dist_{old} - dist_{new})\} \ . \tag{6}$$

where $dist_{old}$ is the Euclidean distance from the current pixel to the destination one, $dist_{new}$ is the Euclidean distance from the selected new pixel to the destination one and a is a real constant, so that $0 \leq a \leq 1$.

As a consequence, the robot will move to the pixel with the highest weight w_i; this operation will be repeated until it arrives to the destination or there is some failure due to a collision with non-detected obstacles. Therefore, the higher factor a is, the bigger differences among the weights w_i exist.

Let us consider now the results of some experiments that will test the suitability of our model. First, Fig. 6 (a) and (b) show the environment in which the robot must wander. At this point, the algorithm developed before is applied; thus, the resulting path map after the initial processing method, using a 3×3 square SE, is depicted in Fig. 6 (c).

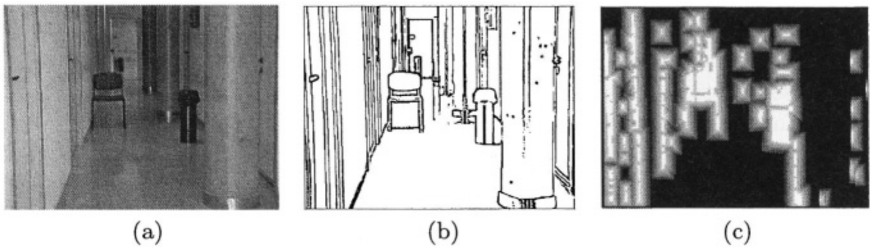


Fig. 6. The environment: (a) Corridor. (b) *MGT* segmentation. (c) Path map

From this map, a trajectory is followed after estimating the weights w_i, using Eq. (6). In Fig. 7 some example paths (where factor a varies from 0.05 to 1.0) are shown.

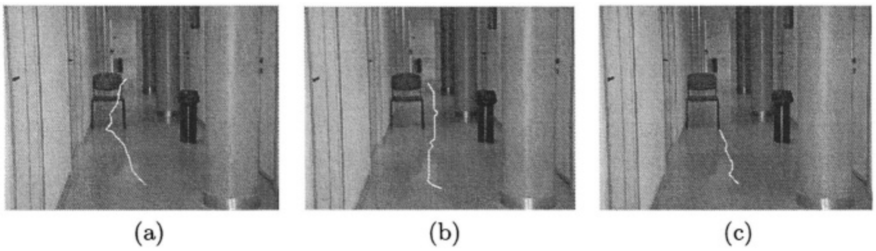


Fig. 7. Some paths followed in the environment: (a) $a = 1.0$. (b) $a = 0.1$. (c) $a = 0.05$

The analysis of these examples shows that if factor a has a high value (Fig. 7 (a)), the algorithm will select a path that easily reaches the destination point, although it is not collision-free as the approaching to the obstacles can be sometimes dangerous. This will lead to the incorporation of some other sensing capabilities to prevent from collision. On the contrary, when factor a has a low value (Fig. 7 (c)), the robot may stop before completing its task, since it is preferred not to move close to the obstacles. As a consequence, factor a has a better behavior when it makes the robot follow an optimal or semi-optimal path that keeps it away from collision (in this example, $a = 0.1$, see Fig. 7 (b)).

7 Conclusions

In this paper, we have described a novel approach to image segmentation based on the selection of a suitable morphological gradient threshold. To do this, global morphological operators have been used to compute the gradient and the Laplacian and, after a proper binarization, the distance between the ideal segmentation and the MGT segmentation has been computed. As a consequence, the gradient threshold with the lowest distance has been selected as the optimal threshold value.

On the other hand, we have developed a vision-based algorithm to generate a path map; starting from an image of the environment, the method is based on the composition of morphological filters. Afterwards, the algorithm has been executed in some environments. The experimentation shows that the main goals of our research task have been accomplished.

As a future work, we propose to extend the results of our research to object classification and recognition. It would be so desirable to consider new simulation experiments with different environments, such as image sequences obtained from a camera placed in a robot platform, where real-time constraints have a great influence in the final recognition results.

References

1. Zomaya, A.Y., Wright, M.R., Nabham, T.M.: On the Path Planning Problem in the Vicinity of Obstacles. Cybernetics & Systems, **29** (1998) 807–868
2. Lin, Z.C., Chow, J.J.: Near Optimal Measuring Sequence Planning and Collision-Free Path Planning with a Dynamic Programming Method. International Journal of Advanced Manufacturing Technology, **18** (2001) 29–43
3. Pal, N.R., Pal, S.K.: A Review on Image Segmentation Techniques. Pattern Recognition, **9** (1993) 1277–1294
4. Arques, P., Compañ, P., Molina, R., Pujol, M., Rizo, R.: A Cybernetic Approach to the Multiscale Minimization of Energy Function: Grey level image segmentation. Kybernetes, **31** (2002) 596–608
5. Arques, P., Compañ, P., Molina, R., Pujol, M., Rizo, R.: Minimization of an Energy Function with Robust Features for Image Segmentation. Kybernetes, **32** (2003) 1481–1491
6. Ouadfel, S., Batouche, M.: MRF-Based Image Segmentation using Ant Colony System. Electronic Letters on Computer Vision and Image Analysis **2** (2003) 12–24
7. Basu, M., Su, M.: Image Smoothing with Exponential Functions. International Journal of Pattern Recognition and Artificial Intelligence, **15** (2001) 735–752
8. Pujol, F., García, J. M., Fuster, A., Pujol, M., Rizo, R.: Use of Mathematical Morphology in Real-Time Path Planning. Kybernetes, **31** (2002) 115–123
9. Pujol, F., Pujol, M., Llorens, F., Rizo, R., García, J. M. : Selection of a Suitable Measurement to Obtain a Quality Segmented Image. In Proc. of the 5th Iberoamerican Symposium on Pattern Recognition, Lisbon, Portugal, (2000) 643–654

Color Image Classification Through Fitting of Implicit Surfaces

Raziel Álvarez, Erik Millán, Ricardo Swain-Oropeza,
and Alejandro Aceves-López

Mecatronics Research Center (CIMe) ITESM-CEM,
Km 3.5 Carretera al Lago de Guadalupe,
Atizapán de Zaragoza, Estado de México, Mexico, 52926
Phone: +52 55 5864 5659
{raziel, emillan, rswain, aaceves}@itesm.mx

Abstract. This paper describes a color classification technique for the color subspaces definition based in 3D reconstruction approaches. These color subspaces use implicit functions to create a bounding surface that will fit a set of characteristic color samples to define a particular color. The implicit subspace reconstruction allow to define clusters of arbitrary shape for a better approximation of the color distribution, reducing misclassification problems obtained when using predefined geometrical shapes. In addition, the proposed method presents less computational complexity than methods based in color signal transformation, allowing dynamical tuning of the subspaces, and provides robustness and ease parameterization.

Keywords: Color classification, computer vision, segmentation, implicit surfaces.

1 Introduction

One of the most common use for a vision system is to provide information to a robot about its environment. The use of a color camera provides a low cost solution compared to devices such as infrared lasers or electromagnetic sensors. However, image analysis requires high computational power, valuable in mobile robots. As in RoboCup [1], a team of autonomous mobile robots are required to play soccer using different colors to identify objects; thus, every image should be rapidly treated using only the processor incorporated into each robot.

An efficient approach to process an image is color segmentation, which consists in classifying the pixels of an image into different color classes. Traditional classification techniques define a cube in the color space for each color class. In this way, a pixel contained within a color cube will be associated to that class. Nevertheless, it is always hard to obtain a cube with most of the pixels associated to a color. Besides, cubes for similar colors may overlap, causing misclassification. In addition, lighting conditions might change dynamically, making an initial classification deficient for a new illumination. In this case, the speed of the algorithm is crucial to provide a fast response to these changes.

In this paper, implicit surfaces are proposed as an alternative representation for classes used in color segmentation. We employ clustering methods to obtain a set of implicit primitives that will be blended to define more precisely a color class. Preliminary results of this work can be seen in [2]. The paper is organized as follows. In section 2, we present some of the existing work on color image segmentation. In section 3, the utility of different surface representations in color classification is evaluated. In section 4 we describe our algorithm in detail. Section 5 shows some examples and results of color image classification using our method. Finally, section 6 discusses the conclusions and further work.

2 Previous Work

The content of an image is encoded through color signals. A *color signal* or *color space* defines a set of attributes that uniquely identify a color. For instance, the RGB color signal identifies a color through its red, green and blue components. The range of component values define the entire range of colors identifiable. In this work, we use the YUV color space, which uses three color components: *Y,* which encodes the luminance for a color, and *U* and *V,* which map its chrominance.

One of the main constraints in artificial vision is the limited processor capacity and fast response required. To improve efficiency, look-up tables are created offline, as depicted in [3], where each classified color is described through a range of values in the color space of the image. Nonetheless, the threshold for each color builds cubical structures that poorly describe the properties of the color, leading to pixel misclassification. One alternative is proposed in [4]. Here, the RGB color space is subdivided in categorical colors, from which membership volumes are generated according to a nearest neighbor criterion. Nevertheless, categorical colors are represented as cubes, leading to the same problem.

Recent work on color segmentation is oriented into transforming a color signal to a different color space, providing an easier classification of color regions than in the original color space. Color signal transformation from YUV to TSL (**T**int, **S**aturation and **L**uminance) is proposed in [5]. This transformation is improved in [6] through genetic algorithms. However, transformations to TSL color space are expensive, as they involve the use of square roots and trigonometric functions.

Another transformation of YUV signal is mentioned in [7], where the destination color space is $\mathrm{Yr}\theta$, transforming the rectangular *U* and *V* coordinates into its polar equivalent r and θ. In the same sense, the method exposed in [8] applies a modification on the RGB space in order to achieve linearly oriented clusters and higher robustness to highlights. Although these approaches offer good results, they require additional computing to transform the image to the new color signal. This is an important drawback that makes their use prohibitive in many real-time applications with reduced computational resources.

Color subspaces might be defined using other geometrical shapes, different to cubes or prisms. In [9], subspaces are defined by linear, tubular distributions that are approximated using ellipsoidal models, defining a static membership function. Nevertheless, these predefined shapes are still prone to the already mentioned problems such as overlapping of color subspaces and misclassification.

In order to solve these problems, we propose to use a 3D reconstruction technique to best fit the shape of a color class in the YUV color space. This technique will use the values corresponding to descriptive pixels for a particular color, forming a cloud of points in the color space. From this cloud, it will construct a shape that bounds tightly the points, leading to a better color classification.

3 Surface Representation

Different representations were evaluated to fit the obtained cloud of samples. A direct representation of the samples distribution can be obtained using a Gaussian model, that will form a properly oriented and scaled hyperellipsoid. This is usually simplified as in [10] by using the squared Mahalanobis distance, which may be calculated using computationally expensive algorithms such as Expectation Maximization or second order algorithms such as gradient descent. This representation works well with few samples and it is suitable for online adaptation of a cluster.

The problem of this parametric technique is that assumes a known underlying distribution, in this case a normal distribution, which have unimodal density. Although color classes may present a roughly normal distribution, they usually have convex and concave discontinuities, even holes, due to the codification of color signals. These discontinuities should be noticed for the final cluster, since they might represent regions of possible misclassification.

Instead, implicit surfaces were chosen, as they can easily differentiate when samples are inside or outside them, providing a natural interface for color classification. In addition, it is easy to blend different surfaces as in Figure 1. In this way, a set of disjoint samples can produce a continuous surface enclosing all the intermediate points from a color class that are rarely found in sampled images.

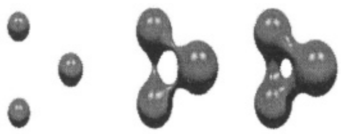


Fig. 1. Spherical implicit surfaces for point skeleton primitives, with different blending parameters

Formally, implicit surfaces, as defined in [11], are two-dimensional, geometric shapes that exist in three dimensional space, defined according to a particular mathematical form. Thus, an implicit surface can be easily characterized by its skeleton and a blending function. The skeleton is a set of primitive elements, such as points and lines, defining individual implicit shapes. The blending function determines the fusion between the individual objects according to their influence. An example of a variation in the blending function is also represented by Figure 1.

Our technique can be compared with nonparametric procedures in terms of their capacity for approximating arbitrary clusters with no underlying known distribution. Two well known approaches are the Nearest-neighbor and the Parzen-window, which can be implemented using a Probabilistic Neural Network, as said in [12]. Although they are powerful methods, they have the drawback of being more complex than the proposed technique, a key point since the desired final application in robotics.

4 Bounding Algorithm

Our approach starts from a set of sample images from which a user selects a set of color samples. A number of primitives is distributed uniformly along the volume of the samples using the k-means algorithm. Once located, the radius of the primitives is obtained from the standard deviation of the samples contained by each primitive. Finally, these primitives are blended to produce the final surface. Figure 2 shows a general overview of this process.

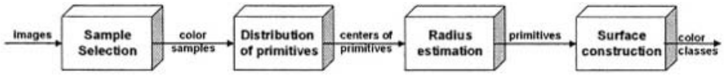


Fig. 2. General description of our approach

4.1 Distribution of the Primitives

One suitable way to perform the data approximation and get the primitives configuration is through a Delaunay Tetrahedronization, as defined in [13], producing a group of tetrahedra that connect all of the samples. Setting a primitive in the center of each tetrahedron and a radius proportional to its volume would yield to a good approximation of the sample data. However, the number of tetrahedra and resulting primitives would be very big, and limiting the number of primitives to the k bigger tetrahedra would result in a poor approximation.

For reconstruction, we will obtain the center and radius of a set of primitives that define an implicit function enclosing the samples. In [14], this reconstruction is obtained by minimizing a cost function, a quadratic sum, representing an error of the desired characteristics in the intended surface. Minimization of this function is neither simple nor fast. Besides, it works directly with samples that lay on the surface and not inside the volume, as the ones used in color classification. A possible solution might be to extract the points that possibly belong to the surface, but many important samples might be lost. Therefore, we propose a different approach to obtain a configuration of primitives that will approximate a better bounding surface by considering the samples inside of it.

To obtain the primitives centers, we use the K-means algorithm. K-means, as defined in [15], is an algorithm for clustering a set of data points into a number of disjoint, non-hierarchical subsets by minimizing a distance criterion. Some options for this criterion are shown in [16]. We selected Euclidean distance, as it describes accurately the color nature in the selected color space; however, it is possible to study different criteria to achieve good results for different applications. K-means is well suited to generate globular clusters, similar to the ones formed by the implicit primitives.

Therefore, we conceive the cloud of samples as a big cluster that will be approximated by a group of small clusters defined by N number of primitives. The K-means algorithm will distribute the primitives in the volume occupied by the samples, then adjust iteratively their position to provide each small cluster with a subset of the samples. These samples will be assigned according to the Euclidean distance criterion. The movement is defined by the following process:

1. Calculate the distance from each sample to the primitive center.
2. Move the centroid to the mean of the samples that are closer to the primitive.

This process guarantees that every sample will belong to a cluster and that the distribution will converge at least to a local minimum. Some of the initially declared N clusters will get no samples; they will be eliminated from the final configuration. The initial condition for this process is that each primitive cluster must have at least two samples. For example, if there are ten clusters and only ten samples, five clusters are discarded to guarantee that, if all five are selected as clusters they will have at least the two necessary samples to form a sphere. If, on the contrary we have more than twice samples than the N number of declared primitives, we only use those N primitives to approximate the surface.

Another issue is the initial location of the primitives, as a bad initialization will converge to poor results. These locations are determined randomly within the bounding box that contains the cloud of samples using a Gaussian distribution, with more primitives around the centroid of the cloud, where we will probably find more samples, and less in the boundaries.

4.2 Estimation of the Radius of the Primitives

For each cluster with at least two samples, the standard deviation for the distance to its samples is calculated. The radius of that primitive is set to a multiple S of the standard deviation, according to the next property: For normally distributed data, there is a relation between the fraction of the included data and the deviation from the mean in terms of standard deviations. Part of this relation is shown in Table 1.

After calculating the radiuses, we obtain a configuration of primitives that fit the samples cloud. However, we may produce primitives with a large radius but just a few samples. These primitives are said to have low density; meaning that the relation between the primitive size and its samples number is below a threshold U. This could lead to a bigger color class than desired, a problem especially considering possible merge with other classes. To diminish this problem we divide each of those primitives in D new primitives, reapplying the algorithm to the samples in the original primitives.

Table 1. Confidence Interval relation

Fraction of data (%)	Number of Standard Deviations from Mean
68.3	1.000
95.4	2.000
99.7	3.000

4.3 Construction of the Surface

The reconstruction scheme is based in [14] which creates an implicit surface composed of a set of spherical primitives, defined by:

$$f_i(P) = \frac{(x - x_{c_i})^2 + (y - y_{c_i})^2 + (z - z_{c_i})^2}{r^2} \tag{1}$$

with the following properties:

$f(P) < T$ For all interior points.
$f(P) > T$ For all exterior points.
$f(P) = T$ For all boundary points.

where T is a scaling parameter of the sphere, and is set to 1 for simplicity.

Primitives placed in various positions with different radiuses will now be blended with a union function to define the final implicit surface. A differentiable algebraic expression proposed by Ricci [17] is used for this union:

$$f = \left\{ \sum_{i=0}^{q} \frac{1}{f_1^n} \right\}^{-\frac{1}{n}} \text{ For } q \text{ primitives} \tag{2}$$

Here, n is referred as the blending degree; when n approaches to infinity, the function converges to its true minimum. Graphically, this means how tight the surface fits the primitives; a large n means tighter, as illustrated in Figure 3.

While the obtained surface will probably not be the tightest possible to the cloud sample, the color segmentation does not demand a high degree of detail; instead, this provides a higher tolerance for omitted samples.

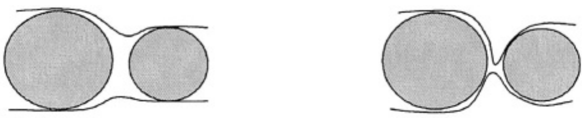


Fig. 3. Blending Degree. Left: Small blending degree, Right: Large blending degree

5 Experimental Results

The algorithm was implemented on a 1.4 MHz Pentium 4 PC with 256 MB memory. The tool works on images or streaming video from an AIBO ERS-210 robot, allowing selecting samples for each color and bound them by an implicit surface. Once the color classes are defined, a look-up table is exported to a text file loaded into a robot for efficient image processing. The configurations produced and the resulting implicit surfaces fit closely the point samples used in the process, producing an accurate representation as depicted in Figure 4.

Fig. 4. Results of configuration of primitives and final implicit surface. Parameters: N=10, D=2,U=2.5, n=2, S=2

We can also modify iteratively the blending degree. Visually it is interpreted as the "blobbiness" of the bounding surface. Figure 5 exemplifies the effect of this parameter. While a smaller blending degree produces higher robustness in color recognition, a large blending degree can produce accurate results. This is useful to solve collisions between different color classes that are close to each other.

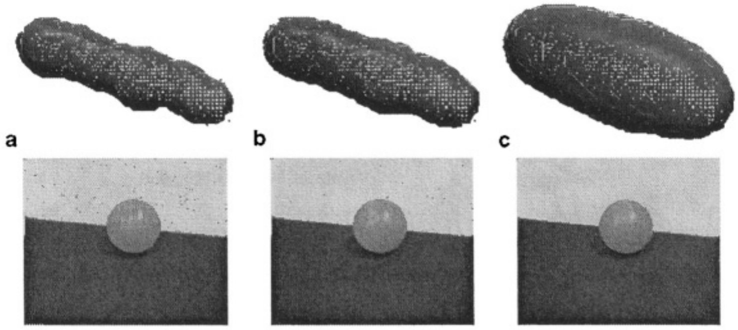


Fig. 5. Bounding surface with different blending degrees and the image segmentation for yellow color in a sample image. (a) n=2. (b) n=1.5. (c) n=1

A comparison between our approach and a traditional color segmentation technique is shown in Figure 6. In both cases, the same color samples were evaluated to segment the image. It is possible to appreciate that the orange ball is partially recognized as yellow when using cubes; besides, much noise produced by the cube segmentation is automatically filtered in our technique.

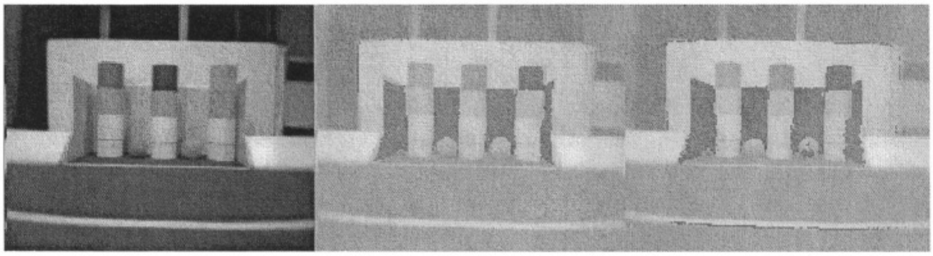


Fig. 6. Color Classification of a yellow color class. Left: Original image. Center: Image segmentation using our approach. Right: Image segmentation using traditional cubes

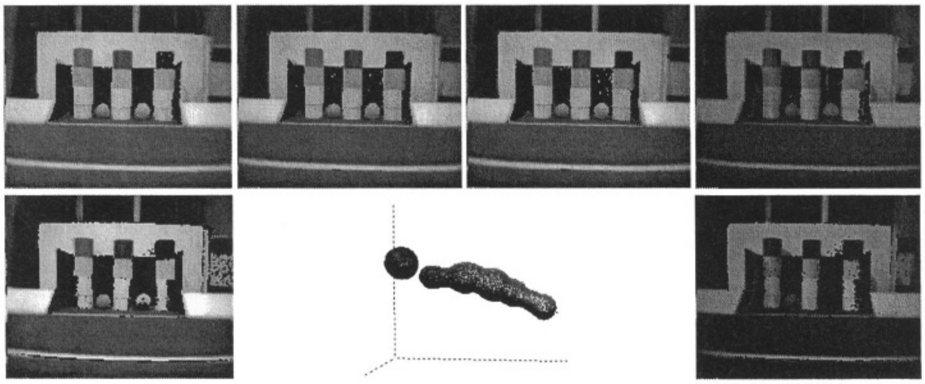


Fig. 7. Robustness to illumination changes. Yellow color is being classified and replaced by blue on the images. Upper row: Image segmentation using our approach with different light intensities. Lower row center: Color subspace used for upper row images. Lower row edges: Image segmentation using traditional cubes on the same images

In Figure 7, a classification is tested in different lighting conditions. The images processed by our algorithm identify colors better even with extreme changes in illumination. This figure also shows the color subspace that is used to identify the color. A traditional approach bounds the samples in this subspace by a cube, leading to misclassifications shown in the lower images. While this tolerance to lighting conditions shows an improvement over traditional techniques, this procedure can be extended to automatically adapt the color subspace dynamically as required by the environment.

In addition, the classification process it is notably fast, as seen in Tables 2 and 3. The speed achieved in the approach permits collecting samples interactively in different light conditions, making possible to obtain feedback on the samples almost immediately.

There are many possible techniques to speed up convergence time and increase precision in the K-means algorithm, as those in [18]. The bottleneck is

Table 2. Time required fitting the surface and reconstructing the look-up table with random initialization for center of primitives. The reconstructed look-up table uses a resolution of 256^3 voxels

Number of samples	100-300	300-500	500-700	700-900	900-1100
Time to find primitives (ms)	23	78	197.1	234.4	287.4
Time to create look-up table (ms)	710.9	849.2	968.3	1113.4	1117.6

Table 3. Time required fitting the surface and reconstructing the look-up table with center initialization based on last adjusted primitives. The reconstructed look-up table uses a resolution of 256^3 voxels

Number of samples	100-300	300-500	500-700	700-900	900-1100
Time to find primitives (ms)	17.1	34.1	43.0	44.1	60.0
Time to create look-up table (ms)	766.1	768.1	840.0	887.2	922.5

the conversion of color classes as a look-up table, due to the complexity of evaluating Equation 2 at a high resolution. Although this long evaluation time is permissible for a static offline use, like our classification tool, it is too expensive in a real-time application. Therefore, some other union equations should be evaluated to avoid this issue. Another solution with little impact in the quality of color segmentation is reducing the resolution of the reconstruction space, originally equal to the size of color space (256^3 for YUV color signal). Approximations using smaller resolutions considerably reduce the reconstruction time.

6 Conclusions and Future Work

A new technique for color classification and image segmentation has been proposed. It presents a good approximation of color subspaces even for color signals that are difficult to classify, like YUV, without the need of transforming the current color signal. The surface of the produced subspace bounds tightly the color samples, reducing merging between color classes, and can easily be adjusted to increase tolerance for a given subspace. The algorithm is fast enough to be used interactively and to obtain feedback on the produced segmentation.

In the future, we will work with overlapping color classes, in order to derive in some criteria to separate the overlapped subspaces. Moreover, we are developing an algorithm that permits us to update color samples during the operation of a vision system with the purpose of reaching higher illumination robustness.

Acknowledgments

This research is part of "Sensor-Based Robotics" project fully supported by NSF-CONACyT grant under 36001-A and by Tec de Monterrey CEM grant under 2167-CCEM-0302-07. Authors thank Bedrich Benes ITESM-CCM and Neil Hernández ITESM-CEM for his comments about this work. This work is part of a MSc Thesis at ITESM-CEM.

References

1. Aceves, A., Junco, M., Ramirez-Uresti, J., Swain-Oropeza, R.: Borregos salvajes 2003. team description. In: RoboCup: 7th Intl. Symp. & Competition. (2003)
2. Junco, M., Ramirez-Uresti, J., Aceves, A., Swain-Oropeza, R.: Tecrams 2004 - mexican team. team description. In: RoboCup: 8th Intl. Symp. & Comp. (2004)
3. Bruce, J., Balch, T., Veloso, M.: Fast and cheap color image segmentation for interactive robots. In: Proceedings of WIRE-2000. (2000)
4. Du, Y., Crisman, J.: A color projection for fast generic target tracking. In: IEEE/RSJ Inter. Conf.on Intelligent Robots and Systems. (1995)
5. Oda, K., Ohashi, T., Kato, T., Katsumi, Y., Ishimura, T.: The kyushu united team in the four legged robot league. In: RoboCup: 6th Intl. Symp. & Comp. (2002)
6. Ingo Dahm, Sebastian Deutsch, M.H., Osterhues, A.: Robust color classification for robot soccer. In: RoboCup: 7th Intl. Symp. & Competition. (2003)
7. Nakamura, T., Ogasawara, T.: On-line visual learning method for color image segmentation and object tracking. In: Proc. of IROS'99. (1999) 222-228
8. Wesolkowski, S., Tominaga, S., Dony, R.D.: Shading and highlight invariant color image segmentation. In: Proc. of SPIE. (2001) 229–240
9. Rasmussen, C., Toyama, K., Hager, G.D.: Tracking objects by color alone. DCS RR-1114, Yale University (1996)
10. Ozyildiz, E., Krahnstoever, N., Sharma, R.: Adaptive texture and color segmentation for tracking moving objects. Pattern Recognition **35** (2002) 2013–2029
11. Bloomenthal, J., Wyvill, B.: Introduction to Implicit Surfaces. Morgan Kaufmann Publishers Inc. (1997)
12. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification (2nd Edition). Wiley-Interscience (2000)
13. Zachmann, G., Langetepe, E.: Geometric data structures for computer graphics. In: Proc. of ACM SIGGRAPH. ACM Transactions of Graphics (2003)
14. Lim, C.T., Turkiyyah, G.M., Ganter, M.A., Storti, D.W.: Implicit reconstruction of solids from cloud point sets. In: Proceedings of the third ACM symposium on Solid modeling and applications, ACM Press (1995) 393–402
15. Bishop, C.M.: Neural networks for pattern recognition. Oxford Univ. Press (1996)
16. Clustan: k-means analysis. http://www.clustan.com/k-means_analysis.html (2004)
17. Ricci, A.: A constructive geometry for computer graphics. The Computer Journal **16** (1973) 157–160
18. Elkan, C.: Using the triangle inequality to accelerate k-means. In: Proc. ICML'03. (2003)

Transforming Fundamental Set of Patterns to a Canonical Form to Improve Pattern Recall

Humberto Sossa, Ricardo Barrón, and Roberto A. Vázquez

Centro de Investigación en Computación-IPN, Av. Juan de Dios Bátiz, esquina con Miguel Othón de Mendizábal, Mexico City, 07738, Mexico

Abstract. Most results (lemmas and theorems) providing conditions under which associative memories are able to perfectly recall patterns of a fundamental set are very restrictive in most practical applications. In this note we describe a simple but effective procedure to transform a fundamental set of patterns (FSP) to a canonical form that fulfils the propositions. This way pattern recall is strongly improved. We provide numerical and real examples to reinforce the proposal.

1 Introduction

The ultimate goal of an associative memory is to correctly recall complete patterns from input patterns that might be altered with additive, subtractive or mixed noise. An associative memory **M** can be viewed as an input-output system as follows: $\mathbf{x} \rightarrow \mathbf{M} \rightarrow \mathbf{y}$, with **x** and **y**, respectively the input and output patterns vectors. Each input vector forms an association with a corresponding output vector, **(x,y)**. For k integer and positive, the corresponding association will be denoted as $\left(\mathbf{x}^k, \mathbf{y}^k\right)$. The associative memory **M** is represented by a matrix whose *ij*-th component is m_{ij}. **M** is generated from a finite a priori set of known associations, known as the *fundamental set of associations,* or simply the *fundamental set* (FS). If ξ is an index, the fundamental set is represented as: $\left\{\left(\mathbf{x}^\xi, \mathbf{y}^\xi\right) \mid \xi = 1,2,\ldots,p\right\}$ with p the cardinality of the set. The patterns that form the fundamental set are called *fundamental patterns.* If it holds that $\mathbf{x}^\xi = \mathbf{y}^\xi \; \forall \xi \in \{1,2,\ldots p\}$, **M** is auto-associative, otherwise it is hetero-associative. A distorted version of a pattern **x** to be recuperated will be denoted as $\tilde{\mathbf{x}}$. If when feeding a distorted version of $\mathbf{x}^w$ with $w \in \{1,2,\ldots,p\}$ to an associative memory **M**, it happens that the output corresponds exactly to the associated pattern $\mathbf{y}^w$, we say that recalling is perfect. Several models for associative memories have emerged in the last 40 years. Refer for example to [1-8].

Most formal propositions providing conditions under which associative memories could be used to perfectly recall patterns of a fundamental set are very restrictive in most practical applications. We propose a simple but effective procedure to transform a fundamental set of patterns (FSP) to a canonical form that fulfils requirements of the propositions. Thus pattern recall is strongly improved. We test our proposal with the recently set of associative memories proposed by Sossa and Barrón in [8]. We provide numerical and real examples to verify the effectiveness of the proposal.

2 Sossa and Barrón's Associative Model

Two kind of associative memories are proposed in [8] to recall real-valued patterns: hetero-associative and auto-associative. Due to space only the hetero-associative case is studied. One hetero-associative memory is described. Let us call HS-memory of type **M**.

Sossa and Barrón's model is founded on the operation of two operations, next defined. Let $P=\left[p_{ij}\right]_{m\times r}$ and $Q=\left[q_{ij}\right]_{r\times n}$ two matrices. The following two matrix operations were defined in [8] to recall real-valued patterns:

1. Operation $\Diamond_{\mathrm{A}}$: $P_{m\times r}\Diamond_{\mathrm{A}}Q_{r\times n}=\left[f_{ij}^{\mathrm{A}}\right]_{m\times n}$ where $f_{ij}^{\mathrm{A}}=\bigotimes_{k=1}^{r}\mathrm{A}\left(p_{ik},q_{kj}\right)$.
2. Operation $\Diamond_{\mathrm{B}}$: $P_{m\times r}\Diamond_{\mathrm{B}}Q_{r\times n}=\left[f_{ij}^{\mathrm{B}}\right]_{m\times n}$ where $f_{ij}^{\mathrm{B}}=\bigotimes_{k=1}^{r}\mathrm{B}\left(p_{ik},q_{kj}\right)$.

According to the operators $\otimes$, A and B used different results can be obtained. If we want, for example, to compensate for additive or subtractive noise, operator $\otimes$ should be replaced either by **max** $(\vee)$ or **min** $(\wedge)$. Median operator **(med)**, should be adopted in the case of mixed noise. In [10] it was shown that the use of this operator provides excellent results in the presence of mixed noise. If $\mathbf{x}\in\mathbf{Z}^n$ and $\mathbf{y}\in\mathbf{Z}^m$, then $\mathbf{y}\Diamond_{\mathrm{A}}\mathbf{x}^t$ is a matrix of dimensions $m\times n$. Relevant simplifications are obtained when operations $\Diamond_{\mathrm{A}}$ and $\Diamond_{\mathrm{B}}$ are applied between vectors:

1. If $\mathbf{x}\in\mathbf{Z}^n$ and $\mathbf{y}\in\mathbf{Z}^m$, then $\mathbf{y}\Diamond_{\mathrm{A}}\mathbf{x}^t$ is a matrix of dimensions $m\times n$, and also it holds that

$$\mathbf{y}\Diamond_{\mathrm{A}}\mathbf{x}^t=\begin{pmatrix}\mathrm{A}(y_1,x_1) & \mathrm{A}(y_1,x_2) & \cdots & \mathrm{A}(y_1,x_n)\\ \mathrm{A}(y_2,x_1) & \mathrm{A}(y_2,x_2) & \cdots & \mathrm{A}(y_2,x_n)\\ \vdots & \vdots & \ddots & \vdots\\ \mathrm{A}(y_m,x_1) & \mathrm{A}(y_m,x_2) & \cdots & \mathrm{A}(y_m,x_n)\end{pmatrix}_{m\times n}.$$

2. If $\mathbf{x}\in\mathbf{Z}^n$ and P a matrix of dimensions $m\times n$, operations $P_{m\times r}\Diamond_{\mathrm{B}}\mathbf{x}$ gives as a result one vector with dimension m, with i-th component given as $\left(P_{m\times r}\Diamond_{\mathrm{B}}\mathbf{x}\right)_i=\operatorname*{med}_{j=1}^{n}\mathrm{B}\left(p_{ij},x_j\right)$.

If $\mathbf{x} \in \mathbf{Z}^n$ and P a matrix of dimensions $m \times n$ then operation $\mathbf{M}_{m\times n} \Diamond_{\mathrm{B}} \mathbf{x}$ outputs an m-dimensional column vector, with i-th component given as: $(\mathbf{M}_{m\times n} \Diamond_{\mathrm{B}} \mathbf{x})_i = \operatorname*{\mathbf{med}}_{j=1}^{n} \mathrm{B}(m_{ij}, x_j)$.

Operators A and B might be chosen among those already proposed in the literature. In this paper we use operators A and B proposed in [6]. Operators A and B are defined as follows:

$$\mathrm{A}(x, y) = x - y \ . \tag{1.a}$$

$$\mathrm{B}(x, y) = x + y \ . \tag{1.b}$$

2.1 Kinds of Noises

The proposed memories can cope with several kinds of noises. Among them: additive, subtractive and mixed. In this paper, we are interested in leading with mixed noise. Let $\mathbf{x} \in \mathbf{R}^n$ an input fundamental pattern to an associative memory. Pattern **x** can be altered or corrupted by mixed noise to produce a vector $\tilde{\mathbf{x}}$ by adding or subtracting at random to each component of **x**, x_i a real c, $\tilde{x}_i = x_i + c$ (additive noise), and $\tilde{x}_i = x_i - c$ (subtractive noise).

2.2 Memory Building and Memory Testing

Two main phases are used to first build and then test the proposed memories. Both phases are next explained.

TRAINING PHASE:

Step 1: For each $\xi = 1, 2, \cdots, p$, from each couple $(\mathbf{x}^\xi, \mathbf{y}^\xi)$ build matrix: $\left[\mathbf{y}^\xi \Diamond_{\mathrm{A}} (\mathbf{x}^\xi)^t\right]_{m\times n}$.

Step 2: Apply the median operator to the matrices obtained in Step 1 to get matrix **M** as follows:

$$\mathbf{M} = \operatorname*{\mathbf{med}}_{\xi=1}^{p} \left[\mathbf{y}^\xi \Diamond_{\mathrm{A}} (\mathbf{x}^\xi)^t\right] . \tag{2}$$

The ij-th component **M** is given as follows:

$$m_{ij} = \operatorname*{\mathbf{med}}_{\xi=1}^{p} \mathrm{A}(y_i^\xi, x_j^\xi) . \tag{3}$$

RECOVERING PHASE:

We have two cases, i.e.:

Case 1: Recall of a fundamental pattern. A pattern $\mathbf{x}^w$, with $w \in \{1, 2, \cdots, p\}$ is presented to the memory **M** and the following operation is done:

$$\mathbf{M}\Diamond_{\mathrm{B}}\mathbf{x}^{w}. \tag{4}$$

The result is a column vector of dimension n, with i-th component given as:

$$\left(\mathbf{M}\Diamond_{\mathrm{B}}x^{w}\right)_i = \operatorname*{med}_{j=1}^{n} \mathrm{B}\left(m_{ij}, x_j^{w}\right). \tag{5}$$

Case 2: Recalling of a pattern from an altered version of it. A pattern $\tilde{\mathbf{x}}$ (altered version of a pattern $\mathbf{x}^{w}$ is presented to the hetero-associative memory **M** and the following operation is done:

$$\mathbf{M}\Diamond_{\mathrm{B}}\tilde{\mathbf{x}}. \tag{6}$$

Again, the result is a column vector of dimension n, with i-th component given as:

$$\left(\mathbf{M}\Diamond_{\mathrm{B}}\tilde{\mathbf{x}}\right)_i = \operatorname*{med}_{j=1}^{n} \mathrm{B}\left(m_{ij}, \tilde{x}_j\right). \tag{7}$$

Sufficient conditions for perfect recall of a pattern of the FS or from altered version of them follow are given by the following results provided in [8]:

Theorem 1. Let $\left\{\left(\mathbf{x}^{\alpha},\mathbf{y}^{\alpha}\right) \mid \alpha = 1,2,\ldots,p\right\}$ with $\mathbf{x}^{\alpha} \in \mathbf{R}^{n}$, $\mathbf{y}^{\alpha} \in \mathbf{R}^{m}$ the fundamental set of an HS-memory **M** and let $\left(\mathbf{x}^{\gamma},\mathbf{y}^{\gamma}\right)$ an arbitrary fundamental couple with $\gamma \in \{1,\cdots,p\}$. If $\operatorname*{med}_{j=1}^{n} \varepsilon_{ij} = 0$, $i=$ 1,...,m, $\varepsilon_{ij} = m_{ij} - \mathrm{A}\left(y_i^{\gamma}, x_j^{\gamma}\right)$ then $\left(\mathbf{M}\Diamond_{\mathrm{B}}\mathbf{x}^{\gamma}\right)_i = \mathbf{y}_i^{\gamma}, i = 1\ldots m$.

More restricted conditions are given by the following:

Corollary 2. Let $\left\{\left(\mathbf{x}^{\alpha},\mathbf{y}^{\alpha}\right) \mid \alpha = 1,2,\ldots,p\right\}$, $\mathbf{x}^{\alpha} \in \mathbf{R}^{n}$, $\mathbf{y}^{\alpha} \in \mathbf{R}^{m}$. A HA-median memory **M** has perfect recall if for all $\alpha = 1,\cdots,p$, $\mathbf{M}^{\alpha} = \mathbf{M}$ where $\mathbf{M} = \mathbf{y}^{\xi}\Diamond_{\mathrm{A}}\left(\mathbf{x}^{\xi}\right)^{t}$ is the associated partial matrix to the fundamental couple $\left(\mathbf{x}^{\alpha},\mathbf{y}^{\alpha}\right)$ and p is the number of couples.

Theorem 3. Let $\left\{\left(\mathbf{x}^{\alpha},\mathbf{y}^{\alpha}\right) \mid \alpha = 1,2,\ldots,p\right\}$, $\mathbf{x}^{\alpha} \in \mathbf{R}^{n}$, $\mathbf{y}^{\alpha} \in \mathbf{R}^{m}$ a FS with perfect recall. Let $\eta^{\alpha} \in \mathbf{R}^{n}$ a pattern of mixed noise. A HA-median memory **M** has perfect recall in the presence of mixed noise if this noise is of median zero, this is if $\operatorname*{med}_{j=1}^{n} \eta_j^{\alpha} = 0, \forall \alpha$.

3 Numerical Examples

In this section we show how the proposed associative memories are able to recall patterns of the FS set and altered versions of them. Only hetero-associative case is studied.

Example 4. Suppose we want to first memorize and then recall the following FS:

$$\mathbf{x}^1=\begin{pmatrix}0.1\\0.0\\0.2\end{pmatrix},\ \mathbf{y}^1=\begin{pmatrix}0.2\\0.3\\0.3\\0.4\end{pmatrix};\ \mathbf{x}^2=\begin{pmatrix}0.4\\0.3\\0.5\end{pmatrix},\ \mathbf{y}^2=\begin{pmatrix}0.5\\0.6\\0.6\\0.7\end{pmatrix}\ \text{and}\ \mathbf{x}^3=\begin{pmatrix}0.7\\0.6\\0.8\end{pmatrix},\ \mathbf{y}^3=\begin{pmatrix}0.8\\0.9\\0.9\\1.0\end{pmatrix}.$$

TRAINING PHASE:

You can easily verify that this FS satisfies Corollary 2, thus:

$$\mathbf{M}=\mathbf{M}^1=\mathbf{M}^2=\mathbf{M}^3=\begin{pmatrix}0.1&0.2&0.0\\0.2&0.3&0.1\\0.2&0.3&0.1\\0.3&0.4&0.2\end{pmatrix}.$$

We should thus expect perfect recall of all the FS. Let us verify this.

RECOVERING PHASE:

$$\mathbf{M}\Diamond_{\mathrm{B}}\mathbf{x}^1=\begin{pmatrix}0.1&0.2&0.0\\0.2&0.3&0.1\\0.2&0.3&0.1\\0.3&0.4&0.2\end{pmatrix}\Diamond_{\mathrm{B}}\begin{pmatrix}0.1\\0.0\\0.2\end{pmatrix}=\begin{pmatrix}\mathbf{med}[\mathrm{B}(0.1,0.1),\mathrm{B}(0.2,0.0),\mathrm{B}(0.0,0.2)]\\\mathbf{med}[\mathrm{B}(0.2,0.1),\mathrm{B}(0.3,0.0),\mathrm{B}(0.1,0.2)]\\\mathbf{med}[\mathrm{B}(0.2,0.1),\mathrm{B}(0.3,0.0),\mathrm{B}(0.1,0.2)]\\\mathbf{med}[\mathrm{B}(0.3,0.1),\mathrm{B}(0.4,0.0),\mathrm{B}(0.2,0.2)]\end{pmatrix}=\begin{pmatrix}\mathbf{med}(0.2,0.2,0.2)\\\mathbf{med}(0.3,0.3,0.3)\\\mathbf{med}(0.3,0.3,0.3)\\\mathbf{med}(0.4,0.4,0.4)\end{pmatrix}=\begin{pmatrix}0.2\\0.3\\0.3\\0.4\end{pmatrix}$$

In the same way:

$$\mathbf{M}\Diamond_{\mathrm{B}}\mathbf{x}^2=\begin{pmatrix}0.1&0.2&0.0\\0.2&0.3&0.1\\0.2&0.3&0.1\\0.3&0.4&0.2\end{pmatrix}\Diamond_{\mathrm{B}}\begin{pmatrix}0.4\\0.3\\0.5\end{pmatrix}=\begin{pmatrix}0.5\\0.6\\0.6\\0.7\end{pmatrix}\ \text{and}\ \mathbf{M}\Diamond_{\mathrm{B}}\mathbf{x}^3=\begin{pmatrix}0.1&0.2&0.0\\0.2&0.3&0.1\\0.2&0.3&0.1\\0.3&0.4&0.2\end{pmatrix}\Diamond_{\mathrm{B}}\begin{pmatrix}0.7\\0.6\\0.8\end{pmatrix}=\begin{pmatrix}0.8\\0.9\\0.9\\1.0\end{pmatrix}$$

Notice how as expected the whole FS has been perfectly recalled. If a FS satisfies the conditions imposed by Theorem 1, and the noise added to a pattern $\mathbf{x}^\alpha$ of this FS satisfies Theorem 3, then no matter the level of noise added, the pattern is perfectly recalled. If this is the case you can easily prove that a given pattern $\mathbf{x}^\alpha$ is recalled through the information of the associated column of matrix $\mathbf{M}\Diamond_{\mathrm{B}}\tilde{\mathbf{x}}^\alpha$ of an element x_i^α of $\mathbf{x}^\alpha$ not affected by noise. Let us verify this with an example.

Example 5. Suppose we want to recall the first fundamental pattern from Example 4 given the following distorted version of its key:

$$\tilde{\mathbf{x}}^1=\begin{pmatrix}0.3\\0.0\\-0.2\end{pmatrix}.$$

As you can appreciate for the distorted pattern, the median of the noise added to **x** equals 0: **med**(0.2,0.0,–0.4)= 0.0 . Also element number two of $\tilde{\mathbf{x}}^1$ is not affected by the added noise. Pattern $\mathbf{x}^1$ is thus recalled through the information of column two (underlined) associated to x_2^1 of $\mathbf{M}\Diamond_B\tilde{\mathbf{x}}^1$. Let us verify this.

RECOVERING PHASE:

$$\mathbf{M}\Diamond_B\mathbf{x}^1=\begin{pmatrix}0.1&0.2&0.0\\0.2&0.3&0.1\\0.2&0.3&0.1\\0.3&0.4&0.2\end{pmatrix}\Diamond_B\begin{pmatrix}0.3\\0.0\\-0.2\end{pmatrix}=\begin{pmatrix}\mathbf{med}[B(0.1,0.3),B(0.2,0.0),B(0.0,-0.2)]\\\mathbf{med}[B(0.2,0.3),B(0.3,0.0),B(0.1,-0.2)]\\\mathbf{med}[B(0.2,0.3),B(0.3,0.0),B(0.1,-0.2)]\\\mathbf{med}[B(0.3,0.3),B(0.4,0.0),B(0.2,-0.2)]\end{pmatrix}=\begin{pmatrix}\mathbf{med}(0.4,\underline{0.2},-0.2)\\\mathbf{med}(0.5,\underline{0.3},-0.1)\\\mathbf{med}(0.5,\underline{0.3},-0.1)\\\mathbf{med}(0.6,\underline{0.4},0.0)\end{pmatrix}=\begin{pmatrix}0.2\\0.3\\0.3\\0.4\end{pmatrix}.$$

4 The Proposal: Case of a General Fundamental Set

In practice most of the fundamental sets of patterns do not satisfy the restricted conditions imposed by Theorem 1 and its Corollary. If this not the case, we propose the following procedure to perfectly recall a general FS. Given a FS not satisfying Theorem 1:

TRAINING PHASE:

Step 1. Transform the FS into an auxiliary fundamental set (FS') satisfying Theorem 1:

```
1) Make $d = x_1^2 - x_1^1$.
2) Make $(\bar{\mathbf{x}}^1,\bar{\mathbf{y}}^1)=(\mathbf{x}^1,\mathbf{y}^1)$.
3) For the remaining couples do {
   For $\xi = 2$ to $p${
       For $i$=1 to $n$ {
           $\bar{x}_i^\xi = \bar{x}_i^{\xi-1} + d$ ; $\hat{x}_i^\xi = \bar{x}_i^\xi - x_i^\xi$ ;
           $\bar{\mathbf{y}}_i^\xi = \bar{\mathbf{y}}_i^{\xi-1} + d$ ; $\hat{\mathbf{y}}_i^\xi = \bar{\mathbf{x}}_i^\xi - \mathbf{x}_i^\xi$.
       }
   }
```

Step 2. Build matrix **M** in terms of set FS': Apply to FS' steps 1 and 2 of the training procedure described at the beginning of Section 2.3.

Remark 6. We can use any *d.* In this work we decided to use however the difference between the first components.

RECOVERING PHASE:

We have also two cases, i.e.:

Case 1: Recalling of a fundamental pattern of FS:

1) Transform $\mathbf{x}^{\xi}$ to $\bar{\mathbf{x}}^{\xi}$ by applying the following transformation: $\bar{\mathbf{x}}^{\xi} = \mathbf{x}^{\xi} + \hat{\mathbf{x}}^{\xi}$.
2) Apply equations (**4**) and (**5**) to each $\bar{\mathbf{x}}^{\xi}$ of FS' to recall $\bar{\mathbf{y}}^{\xi}$.
3) Recall each $\mathbf{y}^{\xi}$ by applying the following inverse transformation: $\mathbf{y}^{\xi} = \bar{\mathbf{y}}^{\xi} - \hat{\mathbf{y}}^{\xi}$.

Case 2: Recovering of a pattern $\mathbf{y}^{\xi}$ from an altered version of its key $\tilde{\mathbf{x}}^{\xi}$:

1) Transform $\tilde{\mathbf{x}}^{\xi}$ to $\bar{\mathbf{x}}^{\xi}$ by applying the following transformation: $\bar{\mathbf{x}}^{\xi} = \tilde{\mathbf{x}}^{\xi} + \hat{\mathbf{x}}^{\xi}$.
2) Apply equations (**6**) and (**7**) to $\bar{\mathbf{x}}^{\xi}$ to get $\bar{\mathbf{y}}^{\xi}$, and
3) Anti-transform $\bar{\mathbf{y}}^{\xi}$ as $\mathbf{y}^{\xi} = \bar{\mathbf{y}}^{\xi} - \hat{\mathbf{y}}^{\xi}$ to get $\mathbf{y}^{\xi}$.

Example 7. Suppose we want to first memorize and then recall the following general fundamental set:

$$\mathbf{x}^1 = \begin{pmatrix} 0.1 \\ 0.0 \\ 0.2 \end{pmatrix}, \mathbf{y}^1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}; \mathbf{x}^2 = \begin{pmatrix} 0.4 \\ 0.2 \\ 0.5 \end{pmatrix}, \mathbf{y}^2 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \text{ and } \mathbf{x}^3 = \begin{pmatrix} 0.6 \\ 0.6 \\ 0.8 \end{pmatrix}, \mathbf{y}^3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}.$$

TRAINING:

1) $d = x_1^2 - x_1^1 = 0.4 - 0.1 = 0.3$.

2) $\bar{\mathbf{x}}^1 = (0.1 \quad 0.0 \quad 0.2)^T$ and $\bar{\mathbf{y}}^1 = (1 \quad 0 \quad 0)^T$.

3) $$\bar{\mathbf{x}}^2 = \begin{pmatrix} 0.4 \\ 0.3 \\ 0.5 \end{pmatrix}, \hat{\mathbf{x}}^2 = \begin{pmatrix} 0.0 \\ 0.1 \\ 0.0 \end{pmatrix}, \bar{\mathbf{y}}^2 = \begin{pmatrix} 1.3 \\ 0.3 \\ 0.3 \end{pmatrix}, \hat{\mathbf{y}}^2 = \begin{pmatrix} 1.3 \\ -0.7 \\ 0.3 \end{pmatrix};$$

$$\bar{\mathbf{x}}^3 = \begin{pmatrix} 0.7 \\ 0.6 \\ 0.8 \end{pmatrix}, \hat{\mathbf{x}}^3 = \begin{pmatrix} 0.1 \\ 0.0 \\ 0.0 \end{pmatrix}, \bar{\mathbf{y}}^3 = \begin{pmatrix} 1.6 \\ 0.6 \\ 0.6 \end{pmatrix}, \hat{\mathbf{y}}^3 = \begin{pmatrix} 1.6 \\ 0.6 \\ -0.4 \end{pmatrix}.$$

You can easily show that: $\mathbf{M} = \mathbf{M}^1 = \mathbf{M}^2 = \mathbf{M}^3 = \begin{pmatrix} 0.9 & 1.0 & 0.8 \\ -0.1 & 0.0 & -0.2 \\ -0.1 & 0.0 & -0.2 \end{pmatrix}$.

RECOVERING PHASE:

Let us consider only Case 2, which is of more interest. Let as suppose we want to recall pattern $\mathbf{y}^2$ from its following distorted key: $\tilde{\mathbf{x}}^2 = (0.6 \quad 0.2 \quad 0.1)^T$.

1) As discussed: $\tilde{\mathbf{x}}^2 = \bar{\mathbf{x}}^2 + \hat{\mathbf{x}}^2 = \begin{pmatrix} 0.6 & 0.3 & 0.1 \end{pmatrix}^T$.

2) $$\mathbf{M}\Diamond_B \tilde{\mathbf{x}}^2 = \begin{pmatrix} 0.9 & 1.0 & 0.8 \\ -0.1 & 0.0 & -0.2 \\ -0.1 & 0.0 & -0.2 \end{pmatrix} \Diamond_B \begin{pmatrix} 0.6 \\ 0.3 \\ 0.1 \end{pmatrix} = \begin{pmatrix} \mathbf{med}(1.5,1.3,0.9) \\ \mathbf{med}(0.5,0.3,-0.1) \\ \mathbf{med}(0.5,0.3,-0.1) \end{pmatrix} = \begin{pmatrix} 1.3 \\ 0.3 \\ 0.3 \end{pmatrix} \cdot$$

Finally, $$\mathbf{y}^2 = \bar{\mathbf{y}}^2 - \hat{\mathbf{y}}^2 = \begin{pmatrix} 1.3 \\ 0.3 \\ 0.3 \end{pmatrix} - \begin{pmatrix} 1.3 \\ -0.7 \\ 0.3 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \cdot$$

5 Experiments with Real Patterns

The proposed associative memories were also tested with real patterns. We used the objects shown in Fig. 1.

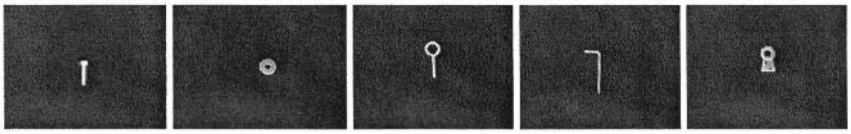


Fig. 1. The five objects used in the experiments. (a) A bolt. (b) A washer. (c) An eyebolt. (d) A hook. (e) A dovetail

5.1 Construction of the Association Matrix

We did not directly recognize the objects by their images. We preferred to do it indirectly by invariant descriptions of each of them. For this, ninety images of each object in different positions, rotations and scale changes were captured. To each image a standard thresholder [9] was applied to get its binary version. Small spurious regions, due to bas thresholding, were eliminated form each image by means of a size filter [10, pp.47-48]. To each of the 19 images of each object (class) the first three Hu geometric invariants, to translations, rotations and scale changes were computed [11]. The five associations were built as:

$$\mathbf{x}^1 = \begin{pmatrix} 0.4429 \\ 0.1594 \\ 0.0058 \end{pmatrix}, \mathbf{y}^1 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}; \mathbf{x}^2 = \begin{pmatrix} 0.1896 \\ 5.78E-5 \\ 4.14E-6 \end{pmatrix}, \mathbf{y}^2 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}; \mathbf{x}^3 = \begin{pmatrix} 0.7038 \\ 0.2911 \\ 0.1825 \end{pmatrix}, \mathbf{y}^3 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix};$$

$$\mathbf{x}^4 = \begin{pmatrix} 1.1421 \\ 1.5517 \\ 0.8467 \end{pmatrix}, \mathbf{y}^4 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}; \mathbf{x}^5 = \begin{pmatrix} 0.2491 \\ 0.0195 \\ 2.41E-5 \end{pmatrix}, \mathbf{y}^5 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix};$$

The three real numbers for each $\mathbf{x}^\alpha$ are the average values of the four Hu invariants computed with each set of 19 images of each object. The "1" at each **y** represents the index of the class of each object. After applying the methodology described in Section 4, matrix **M** is:

$$\mathbf{M} = \begin{pmatrix} 0.5571 & 0.8406 & 0.9942 \\ -0.4429 & -0.1594 & -0.0058 \\ -0.4429 & -0.1594 & -0.0058 \\ -0.4429 & -0.1594 & -0.0058 \\ -0.4429 & -0.1594 & -0.0058 \end{pmatrix}.$$

5.2 Recalling of a Pattern by a Corrupted Version of Its Key

In practical applications the noise added to the values of the patters rarely satisfies Theorem 3. To cope with this situation, we propose the following strategy: Once a general FS has been processed as described in Section 4, one of its patterns is classified in terms of a possible altered version of its key as follows:

$$\mathbf{y}^j, j = \arg\min_i \left(\min_{k=1}^{m} \left(\left| y_k - y_k^i \right| \right) \right). \tag{8}$$

Fifty images (10 for each object), and different from those used to build matrix **M** were used to measure the efficiency of the proposal. Of course the values of the invariants change. Table 1 shows the recalling results. As you can appreciate in 10% of the cases the eyebolt is classified as a bolt, and in 15% of the cases a dovetail is classified as a washer. In remaining cases the objects were correctly classified.

Table 1. Percentage of classification for the test set

	Bolt	Washer	Eyebolt	Hook	Dovetail
Bolt	100%	0	0	0	0
Washer	0	100%	0	0	0
Eyebolt	10%	0	90%	0	0
Hook	0	0	0	100%	0
Dovetail	0	15%	0	0	85%

6 Conclusions

We have described a simple but very efficient procedure to transform a fundamental set of patterns (FSP) to a canonical form that fulfils the most often restricted conditions imposed by formal propositions provided for a special kind of associative memories. We have shown that by doing this, a pattern recall can be strongly improved. We have tested the procedure with the recently set of associative memories proposed by Sossa and Barrón in [8].

Numerical and real examples with images of real objects were provided showing the performance of the improved memories. Results are very promising. We have shown that associative memories combined with invariant features can be used to recognize objects in the presence of image transformations.

Now days, we are: 1) investigating how to use the proposed approach to recognize objects in cluttered environments, and 2) searching for more formal results that tackle the more challenging problem of recognizing pattern without knowing where they come from.

Acknowledgements. This work was supported by CIC-IPN, COFAA-IPN, CONACYT under grants 41529 and 40114Y and CGPI-IPN under grant 20020658.

References

[1] K. Steinbuch, Die Lernmatrix, *Kybernetik,* 1(1):26-45,1961.
[2] D. Wilshaw et al. Non-holographic associative memory, Nature 222:960-962, 1969.
[3] J. A. Anderson, A simple neural network generating an interactive memory, *Mathematical Biosciences,* 14:197-220, 1972.
[4] T. Kohonen, Correlation matrix memories, *IEEE Transactions on Computers,* 21(4):353-359, 1972.
[5] J. J. Hopfield, Neural networks and physical systems with emergent collective computational abilities, *Proceedings of the National Academy of Sciences,* 79: 2554-2558, 1982.
[6] G. X. Ritter el al. Morphological associative memories, *IEEE Transactions on Neural Networks,* 9:281-293, 1998.
[7] C. Yáñez, Associative Memories based on Order Relations and Binary Operators (In Spanish), PhD Thesis, Center for Computing Research, February of 2002.
[8] H. Sossa and Ricardo Barrón. New associative model for pattern recall in the presence of mixed noise, IASTED Fifth International Conference on Signal and Image Processing (SIP 2003), August 13-15, 2003, Honolulu, Hawaii, USA.
[9] N. Otsu, A threshold selection method from gray-level histograms, *IEEE Transactions on SMC,* 9(1): 62-66,1979.
[10] R. Jain et al. *Machine Vision,* McGraw-Hill, 1995.
[11] M. K. Hu, Visual pattern recognition by moment invariants, IRE Transactions on Information Theory, 8: 179-187,1962.

Nonlinear System Identification Using ANFIS Based on Emotional Learning

Mahdi Jalili-Kharaajoo

Young Researchers Club, Islamic Azad University,
Tehran, Iran
mahdijalili@ece.ut.ac.ir

Abstract. In this paper using emotional learning, the performance of Adaptive Network Fuzzy Inference System (ANFIS) will be enhanced. Neural networks and Neurofuzzy models have been successfully used in nonlinear time series prediction. Many of the existing methods for learning algorithm of these networks, constructed over Takagi Sugeno fuzzy inference system, are characterized by high generalization. However, they differ in computational complexity. A practical approach towards the prediction of real world data such as the sunspot number time series profound a method to follow multiple goals. For example predicting the peaks of sunspot numbers (maximum of solar activity) is more important due to its major effects on earth and satellites. The proposed Emotional Learning Based Fuzzy Inference System (ELFIS) has the advantage of low computational complexity in comparison with other multi-objective optimization methods.

1 Introduction

Predicting the future has been an interesting important problem in human mind. Alongside great achievements in this endeavor there remain many natural phenomena the successful predictions of which have so far eluded researchers. Some have been proven unpredictable due to the nature of their stochasticity. Others have been shown to be chaotic: with continuous and bounded frequency spectrum resembling white noise and sensitivity to initial conditions attested via positive Lyapunov exponents resulting in long term unpredictability of the time series. There are several developed methods to distinguish chaotic systems from the others, however model-free nonlinear predictors can be used in most cases without changes.

Comparing with the early days of using classical methods like polynomial approximators, neural networks have shown better performance, and even better are their successors: Neurofuzzy models [1], [2], [11], [19],[20]. Some remarkable algorithms have been proposed to train the neurofuzzy models [1], [7], [8], [10], [11], [14]. The pioneers, Takagi and Sugeno, presented an adaptive algorithm for their fuzzy inference system [14]. The other methods include adaptive B-spline modeling [8] and adaptive network-based FIS [7]. These learning methods fulfill the principle of network parsimony which leads to high generalization performance in predictions or estimations. The parsimony principle says that the best models are those with the simplest acceptable structures and smallest number of adjustable parameters.

The next generation of neurofuzzy networks includes locally linear models which are piecewise linear approximations of the nonlinear function and can be interpreted as extensions of radial basis function networks [10], [11], [2]. In recent years RBF networks have shown good results in various fields of prediction and estimation; therefore, one can expect good results in predicting by locally linear neurofuzzy models. Of course these models have noticeable characteristics like low prediction errors, relatively low computational complexity and high generalization.

Following the directions of biologically motivated intelligent computing, the emotional learning method has been introduced as a kind of reinforcement learning. This approach is based on an emotional signal which shows the emotions of a critic about the overall performance of the system. The distinctive feature of emotional signal is that it can be produced by any combination of objectives or goals which improve the estimation or prediction. The loss function will be defined just as a function of emotional signal and the training algorithm will be simply designed to minimize this loss function. So the model will be trained to provide the desired performance in a holistic manner.

The emotional learning algorithm is a model-free method which has three distinctive properties in comparison with other neurofuzzy learning algorithms. For one thing, one can use very complicated definitions for emotional signal without increasing the computational complexity of algorithm or worrying about differentiability or renderability into recursive formulation problems. For another, the parameters can be adjusted in a simple intuitive way to obtain the best performance. Besides, the training is very fast and efficient. As can be seen these properties make the method preferable in real time applications like control tasks, as have been presented in literature [3], [4], [5], [9], [12], [18].

In this research the emotional learning algorithm has been used in predicting some real world time series: the sunspot number and the price of securities. The main advantage of introducing an emotional signal in prediction is adjusting some important features. For example, in predicting the sunspot number the peak points which are related to the maximum of eleven-year cycle of solar activity are more important than the others due to their strong effects on space weather, earth and satellites. Therefore, in order to achieve good predictions in these points the error and delay of predicting the peaks may be included in the definition of emotional signal. Additional achievements are fast training of model and low computational complexity.

This method can be used in various forms. A multi objective prediction can be done by introducing a simple definition of emotional signal at the first steps of training and exchanging it with a more accurate one to improve the accuracy in important regions. As an interpretation at the first stage of prediction the critic will just give attention to the overall performance of prediction and after some time, when an acceptable accuracy is obtained, it will incline to more important goals.

The main contribution of this paper is to provide accurate predictions using emotional learning algorithm in Takagi Sugeno neurofuzzy model. The results are compared with other neural and neurofuzzy models like RBF network and ANFIS based on the prediction error in important regions and computational complexity.

2 Neurofuzzy Models

Two major approaches of trainable neurofuzzy models can be distinguished. The network based Takagi-Sugeno fuzzy inference system and the locally linear neurofuzzy model. The mathematical description of these models will be described in this section. It is easy to see that the locally linear model is equivalent to Takagi-Sugeno fuzzy model under certain conditions, and can be interpreted as an extension of normalized RBF network as well.

The Takagi-Sugeno fuzzy inference system is based on fuzzy rules of the following type

$$\begin{aligned}&\textit{Rule}_i: \textit{ If } u_1 = A_{i1} \textit{ And} \dots \textit{And } u_p = A_{ip}\\ &\textit{then } \hat{y} = f_i(u_1, u_2, \dots, u_p)\end{aligned} \tag{1}$$

where $i = 1...M$ and M is the number of fuzzy rules. $u_1,...,u_p$ are the inputs of network, each A_{ij} denotes the fuzzy set for input u_j in rule i and $f_i(.)$ is a crisp function which is defined as a linear combination of inputs in most applications

$$\hat{y} = \omega_{i0} + \omega_{i1}u_1 + \omega_{i2}u_2 + \dots + \omega_{ip}u_p \tag{2}$$

Matrix form $\hat{y} = a^T(\underline{u}) \cdot W$

Thus, the output of this model can be calculated by

$$\hat{y} = \frac{\sum_{i=1}^{M} f_i(\underline{u})\mu_i(\underline{u})}{\sum_{i=1}^{M} \mu_i(\underline{u})} \quad ; \quad \mu_i(\underline{u}) = \prod_{j=1}^{p} \mu_{ij}(u_j) \tag{3}$$

where $\mu_{ij}(u_j)$ is the membership function of j^0 input in the i^{th} rule and $\mu_i(\underline{u})$ is the degree of validity of the i^{th} rule.

This system can be formulated in the basis function realization which leads to relation between Takagi-Sugeno fuzzy model and normalized RBF network. The basis function will be

$$\phi_i(\underline{u}) = \frac{\mu_i(\underline{u})}{\sum_{j=1}^{M} \mu_j(\underline{u})} \tag{4}$$

as a result

$$\sum_{j=1}^{M} \phi_j(\underline{u}) = 1 \tag{5}$$

This neurofuzzy model has two sets of adjustable parameters; first the antecedent parameters, which belong to the input membership functions such as centers and deviations of Gaussians; second the rule consequent parameters such as the linear weights of output in equation (2). It is more common to optimize only the rule consequent parameters. This can be simply done by linear techniques like least squares [1]. A linguistic interpretation to determine the antecedent parameters is usually adequate. However, one can opt to use a more powerful nonlinear optimization method to optimize all parameters together.

Gradient based learning algorithms can be used in the optimization of consequent linear parameters. Supervised learning is aimed to minimize the following loss function (mean square error of estimation):

$$J = \frac{1}{N}\sum_{i=1}^{N}\left(y(i) - \hat{y}(i)\right)^2 \tag{6}$$

where N is the number of data samples.

According to the matrix form of (2) this loss function can be expanded in the quadratic form

$$J = W^T R W - 2W^T P + Y^T Y/N \tag{7}$$

Where $R = (1/N)A^T A$ is the autocorrelation matrix, A is the $N \times p$ solution matrix whose *ith* row is $a(\underline{u}(i))$ and $P = (1/N)A^T y$ is the p dimensional cross correlation vector.

From

$$\frac{\partial J}{\partial W} = 2RW - 2P = 0 \tag{8}$$

The following linear equations are obtained to minimize *J:*

$$RW = P \tag{9}$$

and W is simply defined by pseudo inverse calculation.

One of the simplest local nonlinear optimization techniques is the steepest descent. In this method the direction of changes in parameters will be opposite to the gradient of cost function

$$\Delta W(i) = -\frac{\partial J}{\partial W(i)} = 2P - 2RW(i) \tag{10}$$

and

$$W(i+1) = W(i) + \eta \cdot \Delta W(i) \tag{11}$$

where η is the learning rate.

Other nonlinear local optimization techniques can be used in this way, e.g. the conjugate gradient or Levenberg-Marquardt which are faster than steepest descent. All these methods have the possibility of getting stuck at local minima.

Some of the advanced learning algorithms that have been proposed for the optimization of parameters in Takagi-Sugeno fuzzy inference system include ASMOD (Adaptive spline modeling of observation data) [8], ANFIS (Adaptive network based fuzzy inference system) [7] and FUREGA (fuzzy rule extraction by genetic algorithm) [11].

ANFIS is one of the most popular algorithms that has been used for different purposes, such as system identification, control, prediction and signal processing. It is a hybrid learning method based on gradient descent and least square estimation.

The ASMOD algorithm is an additive constructive method based on k-d tree partitioning. It reduces the problems of derivative computation, because of the favorable properties of B-spline basis functions. Although ASMOD has a complicated procedure, it has advantages like high generalization and accurate estimation.

One of the most important problems in learning is the prevention of overfitness. It can be done by observing the error index of test data at each iteration. The learning algorithm will be terminated, when the error index of test data starts to increase.

3 Emotional Learning

Most of new learning algorithms like reinforcement learning, Q-learning and the method of temporal differences are characterized by their fast computation and in some cases lower error in comparison with the classical learning methods. Fast training is a notable consideration in some control applications. However, in prediction applications, two more desired characteristics of a good predictor are accuracy and low computational complexity.

The Emotional learning method is a psychologically motivated algorithm which is developed to reduce the complexity of computations in prediction problems. Using the distinctive properties of this method one can follow multiple goals in prediction. In this method the reinforcement signal is replaced by an emotional cue, which can be interpreted as a cognitive assessment of the present state in light of goals and intentions. The main reason of using emotion in a prediction problem is to lower the prediction error in some regions or according to some features. For example predicting the sunspot number is more important in the peak points of the eleven-year cycle of sun activity, or predicting the price of securities with better approximation of variance may be desired. Of course these objectives can be pursued by other methods too, but by emotional learning any complicated multi objective problem can be solved using a fast, efficient computation.

This method is based on an emotional signal which shows the emotions of a critic about the overall performance of predictor. The distinctive feature of emotional signal is that it can be produced by any combination of objectives or goals which improve estimation or prediction. The loss function will be defined just as a function of emotional signal and the training algorithm will be simply designed to decrease this loss function. So the predictor will be trained to provide the desired performance in a holistic manner. If the critic emphasizes on some regions or some properties, this can be observed in his emotions and simply affects the characteristics of predictor.

Thus the definition of emotional signal is absolutely problem dependent. It can be a function of error, rate of error change and many other features. Then a loss function is defined based on the emotional signal. A simple form can be

$$J = \frac{1}{2} K \sum_{i=1}^{N} es(i)^2 \tag{12}$$

where es is the emotional signal.

Learning is adjusting the weights of model by means of a nonlinear optimization method, e.g. the steepest descent or conjugate gradient.

With steepest descent method the weights will be adjusted by the following variations:

$$\Delta\omega = -\eta \frac{\partial J}{\partial \omega} \tag{13}$$

where η is the learning rate of the corresponding neurofuzzy controller and the right hand side can be calculated by chain rule:

$$\frac{\partial J}{\partial \omega} = \frac{\partial J}{\partial es} \cdot \frac{\partial es}{\partial y} \cdot \frac{\partial y}{\partial \omega} \tag{14}$$

According to (12): $\frac{\partial J}{\partial es} = K.es$

and $\frac{\partial y}{\partial \omega}$ is accessible from (3) where $f_i(.)$ is a linear function of weights.

Calculating the remaining part, $\frac{\partial es}{\partial y}$, is not straightforward in most cases. This is the price to be paid for the freedom to choose any desired emotional cue as well as not having to impose presuppose any predefined model. However, it can be approximated via simplifying assumptions. If, for example error is defined by

$$e = y_r - y \tag{15}$$

where y_r is the output to be estimated, then

$$\frac{\partial es}{\partial y} = -\frac{\partial es}{\partial e} \tag{16}$$

can be replaced by its sign (-1) in (14). The algorithm is after all, supposed to be satisficing rather than optimizing.

Finally the weights will be updated by the following formula:

$$\Delta\omega = -K \cdot \eta \cdot es \cdot \frac{\partial y}{\partial \omega} = -K \cdot \eta \cdot es \cdot \frac{\sum_{i=1}^{M} u_i \mu_i(\underline{u})}{\sum_{i=1}^{M} \mu_i(\underline{u})} \tag{17}$$

The definition of emotional signal and the gradient based optimization of the emotional learning algorithm in neurofuzzy predictors is clarified among two examples in next section.

4 Predicting the Sunspot Number

Solar activity has major effects not only on satellites and space missions but also on communications and weather on earth. This activity level changes with a period of eleven years, called solar cycle. The solar cycle consists of an active part, the solar maximum, and a quiet part, solar minimum. During the solar maximum there are many sunspots, solar flares and coronal mass ejections. A useful measure of solar activity is the observed sunspot number. Sunspots are dark blemishes on the face of sun and last for several days. The SESC sunspot number is computed according to the Wolf sunspot number $R=k(10g+s)$, where g is the number of sunspot groups, s is the total number of spots in all the groups and k is a variable scaling factor that indicates the conditions of observation.

A variety of techniques have been used in the prediction of solar activity and its effects by sunspot number. The sunspot number shows low dimensional chaotic behavior and its prediction is a challenging problem for researchers. However, good results are obtained by methods proposed in several articles [6], [13], [15], [16], [17], [19], [20].

In this research, both the monthly and the yearly averaged sunspot number are used to predict. Fig.1 shows the history of solar cycles based on yearly sunspot numbers.

The error index in predicting the sunspot number in this article, the normalized mean square error (NMSE), is defined as follow

$$NMSE=\left(\frac{\sum_{i=1}^{n}(y-\hat{y})^2}{\sum_{i=1}^{n}(y-\bar{y})^2}\right) \tag{18}$$

In which $y, \hat{y}$ *and* $\bar{y}$ are observed data, predicted data and the average of observed data respectively.

At first the emotional learning algorithm has been used to enhance the performance and accuracy of a neurofuzzy predictor based on ANFIS. The emotional signal is computed by a linguistic fuzzy inference system with error and rate of error change as inputs. Figure 2 presents the target and predicted outputs of the test set (from 1920 to 2000). The lower diagram shows the results of best fitted data by ANFIS. The training is done with optimal number of fuzzy rules and epochs (74 epochs) and has been continued until the error of test set had been started to increase. The other diagram shows the target and predicted values after using emotional learning. The emotional algorithm is just used in fine tuning of the weights of neurofuzzy model which has been initiated and adjusted by ANFIS. The error index, NMSE, has been decreased from 0.1429 to 0.0853 after using emotional learning. It's interesting that training ANFIS to the optimum performance takes approximately ten times more computation effort than the emotional learning to improve the prediction. Thus combining ANFIS with the emotional learning is a fast efficient method to improve the quality of predictions, at least in this example.

The next results are reported as a comparison of the quality of predicting the monthly sunspot number by emotional learning with other learning methods, especially the RBF network and ANFIS. All methods are used in their optimal performance. Over fitness is prevented by observing the mean square error of test data during training. Three regressors are used as the inputs of models. The specifications of methods, NMSE of predictions and computation times (on a 533 MHz Celeron processor) are presented in Table 1. Note that ELFIS is an abbreviation of the proposed predictor for Emotional Learning based Fuzzy Inference System.

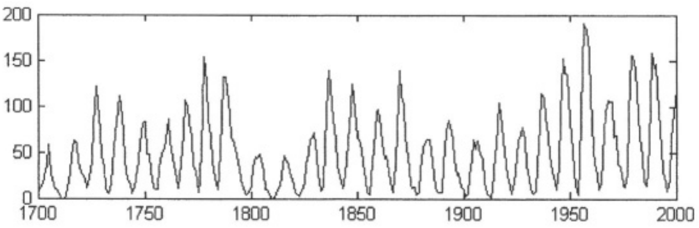


Fig. 1. The Yearly Averaged Sunspot Number

Table 1. Comparison of Predictions by Selected Neural and Neurofuzzy Models

	Specifications	Computation Time	NMSE
ANFIS	8 rules and 165 epochs	89.5790 sec.	0.1702
RBF	7 neurons in hidden layer	84.7820 sec.	0.1314
ELFIS	3 Sugeno type fuzzy rules	22.3320 sec.	0.1386

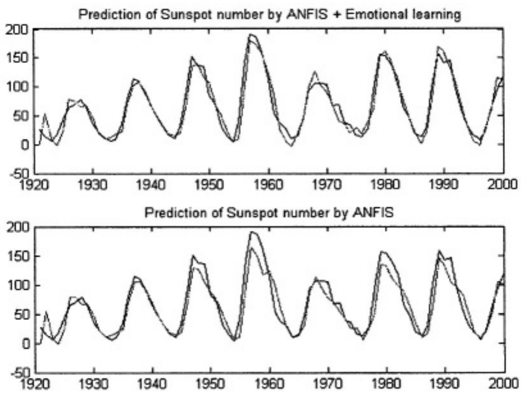


Fig. 2. Enhancement in the Prediction of Sunspot Sumber by Emotional Learning, Applied to ANFIS

ELFIS is based on Takagi Sugeno fuzzy inference system, the emotional signal is computed by a fuzzy critic whose linguistic rules are defined by means of weighted error, rate of error change and the last targeted output. Thus the critic shows exaggerated emotions in the solar maximum regions, especially the peak points, where the prediction is more important due to its effects on earth and satellites. The weights of neurofuzzy model (equation 2) are adjusted by emotional learning via 17.

According to Table 1 learning in ELFIS is at least four times faster than the others and is more accurate than ANFIS. It is remarkable that using a functional description of emotional signal rather than the fuzzy description will generate faster algorithm, But finding such a suitable function is not easy.

Figures 3 to 5 show the predictions by RBF network, ANFIS and ELFIS respectively. These diagrams are a part of test set, especially the cycle 19 which has an above average peak in 1957. It's observable that ELFIS generates the most accurate prediction in the solar maximum; however the NMSE of RBF is the least. Noticeably it's more important to predict the peak points with small errors rather than the points in minimum regions. This is a result of the emotions of critic in the solar maximum.

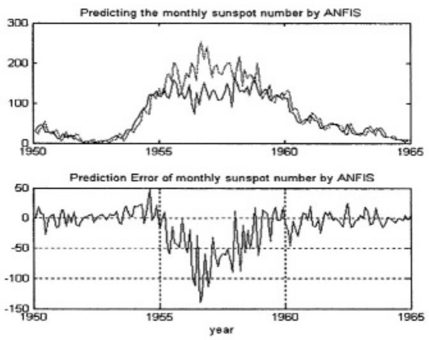


Fig. 3. Predicting the Sunspot Number by ANFIS

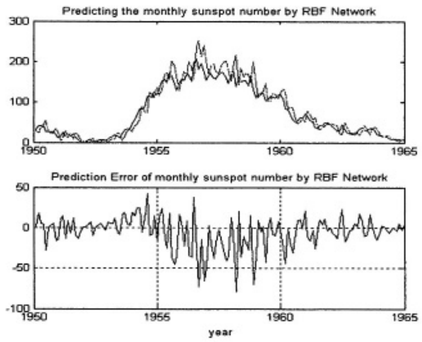


Fig. 4. Predicting the Sunspot Number by RBF Network

RBF network generates more accurate prediction through the test set; however it is observed that emotional learning provides better results in the solar maximum, especially at the above-average peak of 1957. Other test sets are chosen to stop the training in order to avoid over fitness. In this case even better NMSE can be obtained by RBF, in expense of higher prediction error especially in 1957.

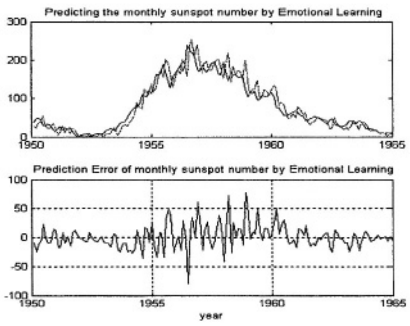


Fig. 5. Predicting the Sunspot Number by Emotional Learning Based Fuzzy Inference System

5 Conclusion

In this paper, the proposed emotional learning based fuzzy inference system (ELFIS) has been used as a predictor in the prediction of solar activity (the sunspot number time series) the emotional signal is determined with emphasis on the solar maximum regions (the peak points of sunspot number) and it has shown better results in comparison with RBF network and ANFIS. The definition of emotional signal is an important aid in emotional learning algorithms, which provides high degrees of freedom. In the problem of security price prediction, better performance can be obtained through the use of variables in addition to the lagged values of the process to be predicted (e.g. fundamentalist as well as chartist data).

References

1. Brown M., Harris C.J. (1994), Neuro fuzzy adaptive modeling and control, Prentice Hall, New York.
2. Eppler W., Beck H.N. (1999), "Peicewise linear networks (PLN) for function approximation," Proc. of IEEE Int. Con. on neural networks, Washington.
3. Fatourechi M., Lucas C., Khaki Sedigh A. (2001), "An Agent-based Approach to Multivariable Control," Proc. of IASTED International Conference on Artificial Intelligence and Applications, Marbella, Spain, pp. 376-381.
4. Fatourechi M., Lucas C., Khaki Sedigh A. (2001) "Reducing Control Effort by means of Emotional Learning," Proceedings of 9th Iranian Conference on Electrical Engineering, (ICEE2001), pp. 41-1 to 41-8, Tehran, Iran.
5. Fatourechi M., Lucas C, Khaki Sedigh A. (2001) "Reduction of Maximum Overshoot by means of Emotional Learning," Proceedings of 6th Annual CSI Computer Conference, Isfahan, Iran, pp. 460-467.
6. Izeman A. J. (1985), "Wolf J.R. and the Zurich sunspot relative numbers," The Mathematical Intelligence, Vol.7, No.1, pp. 27-33.
7. Jang J.R. (1993) "ANFIS: Adaptive network based fuzzy inference system," IEEE Tran. On systems, Man and Cybernetics, 23(3), pp. 665-685.
8. Kavli T. (1993), "ASMOD: An algorithm for adaptive spline modeling of observation data," Int. J. of Control, 58(4), pp. 947-967.
9. Lucas C., Jazbi S.A., Fatourechi M., Farshad M. (2000) "Cognitive Action Selection with Neurocontrollers," 3rd Irano-Armenian Workshop on Neural Networks, Armenia.
10. Nelles O. (1997) "Orthonormal basis functions for nonlinear system identification with local linear model trees (LOLIMOT)", IFAC symposium for system identification (SYSID), Fukuda, Japan, pp. 667-672.
11. Nelles O. (2001), Nonlinear system identification, Springer Verlag, Berlin.
12. Perlovsky L.I. (1999), "Emotions, Learning and control," proc. of IEEE Int. symp. On Intelligent control/Intelligent systems and semiotics, Cambridge MA, pp. 132-137.
13. Schatten K.H., Pesnell W.D. (1993), "An early solar dynamo prediction: Cycle 23 ~ Cycle 22," Geophysical research letters, 20, pp. 2257-2278.
14. Takagi T., Sugeno M. (1985), "Fuzzy identification of systems and its applications to modeling and control", IEEE Tran. On systems, Man and Cybernetics, 15, pp. 116-132.
15. Thompson R.J. (1993), "A technique for predicting the amplitude of the solar cycle", Solar physics, pp. 148, 383.

16. Tong H., Lim K. (1980), “Threshold Autoregressive limit cycles and cyclical data,” J. Roy. Statistics. Soc. B, no.42, pp. 245-292.
17. H. Tong (1996), Nonlinear time series: A dynamical system approach, Oxford press, UK.
18. Ventura R., Pinto Ferreira C. (1999), “Emotion based control systems,” proc. of IEEE Int. symp. On Intelligent Con./Intelligent Sys., Cambridge MA, pp. 64-66.
19. Weigend A., Huberman B., Rumelhart D.E. (1990), “Predicting the future: a connectionist approach,” Int. J. Of Neural systems, vol. 1, pp. 193-209.
20. Weigend A., Huberman B., Rumelhart D.E., (1992), “Predicting sunspots and exchange rates with connectionist networks,” in Nonlinear Modeling and Forecasting, Casdagli, Eubank: Editors, Addison-Wesley, pp. 395-432.

Improving k-NN by Using Fuzzy Similarity Functions

Carlos Morell, Rafael Bello, and Ricardo Grau

Department of Computer Science,
Central University of Las Villas, Cuba
{cmorellp, rbellop, rgrau}@uclv.edu.cu

Abstract. The k-Nearest Neighbor (k-NN) is the basis for many lazy learning algorithms. It uses a similarity function to generate predictions from stored instances. Many authors have shown that the performance of k-NN is highly sensitive to the definition of its metric similarity. In this paper we propose the use of the fuzzy set theory in the definition of similarity functions. We present experiments with a real application to demonstrate the usefulness of this approach.

1 Introduction

Instance-Based Learning (IBL) known also as Case-based or Memory-based Learning algorithms, unlike other machine learning techniques, do not construct abstract hypotheses but, instead, base their answers on similarities to specific training instances. Training is usually very simple: it just stores the training instances. Generalization is postponed until a request for a new instance is received. Therefore, these methods are also called lazy.

Current IBL algorithms assume that cases are described using a feature-value representation, where features are either predictors or goal features. An instance $c=(c_1,c_2,\ldots,c_n,c_g)$ consists of n-describing features and a goal feature. Given a new query $q=(q_1,q_2,\ldots,q_n)$, k-most-similar cases are retrieved to predict the goal feature q_g. this similarity *sim* between query *q* and any instance *c* is computed by the following expression:

$$sim(q,c)=\sum_{a=1}^{n} w_a \cdot sim_a(q_a,c_a) \tag{1}$$

where w_a is the weight for feature *a* and sim_a is the local similarity measure for feature *a*. In general, it is assumed that $\sum_{a=1}^{n} w_a = 1$ and $sim_a(q_a,c_a)\in[0,1]$, for all attributes, yielding the similarity between a query and a case $sim(q,c)\in[0,1]$.

Local similarity measure sim_a is usually defined by the expression:

$$sim_a(q_a,c_a)=\begin{cases} 1-|q_a-c_a| & \text{if } a \text{ is numeric} \\ 1 & \text{if } q_a=c_a \\ 0 & \text{if } q_a\neq c_a \end{cases} \tag{2}$$

This algorithm predicts that the goal value is the similarity-weighted average derived from the k-most-similar instances. If the goal feature is symbolic-valued then it is called a classification task. The class of the query vector is the most probable class calculated from the expression:

$$p_{q,t} = \frac{\sum_{r \in K} \delta_{r,t} \cdot sim(q,r)^2}{\sum_{r \in K} sim(q,r)^2} \tag{3}$$

where $\delta_{r,t} = \begin{cases} 1 & \text{if } r_g = t \\ 0 & \text{if } r_g \neq t \end{cases}$

If the goal feature is numeric-valued the target value is calculated from:

$$q_g = \frac{\sum_{r \in K} r_g \cdot sim(q,r)^2}{\sum_{r \in K} sim(q,r)^2} \tag{4}$$

There are several directions to improve the k-NN performance. In this paper we define several (local and global) similarity measures by using linguistic variables to deal with numeric features.

2 Modeling Numeric Variables as Linguistic Variables

Sometimes local metrics as defined by equation (2) are not satisfactory. In that case, one can develop specialized ones to account for relationships existing among feature values. For example, consider a diagnosis system for human diseases where a typical feature is the body temperature. Additionally, consider a query q containing the value 38.2° at feature temperature (q_t=38.2) and two instances containing x_t=37.7 and y_t=39. Equation (2) tells us that x_t is more similar to q_t than y_t while an expert can tell us that 38.2 and 39 are more similar because the two temperatures can be considered as medium fever.

To model numeric variables as linguistic variables, the term set where the linguistic variable takes values, must be defined. In the example below the term set can be defined as follows: T(fever)={low, medium, high}. Each linguistic term is associated with a fuzzy set, each of which has a defined membership function (MF). Formally, a fuzzy set A in U is expressed as a set of ordered pairs $A = \{ (x, \mu_A(x)) \mid x \text{ in } U \}$ where $\mu_A(x)$ is the membership function that gives the degree of membership of x. This indicates the degree to which x belongs in set A. The figure below illustrates a linguistic variable fever with tree associated linguistic terms: namely, "low", "medium", and "high". Each of these linguistic terms is associated to a fuzzy set defined by a corresponding membership function.

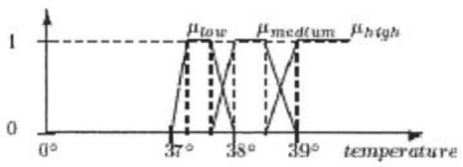


Fig. 1. Fuzzy sets representing the concept "fever"

There are several reasons for using fuzzy linguistic variables instead of other types of attributes, among them: It is easier to build objects using linguistic terms than to use the values of the universe of the feature, see [3] and [4]; it is necessary to consider the domain of the feature as a finite set in spite of its being the universe of an infinite set, see [5]; the use of linguistic terms allows the situation represented by the object to be seen clearer, see [6]; it favors a reduction of the number of cases in the base, see [6]; the relationship between the value of predictive features (those that take value upon describing the problem) and the objective features (whose values we wish to determine) can be established easily, see [7]; the fuzzy approach provides techniques to deal with imprecision, see [8], [9], [10], [11] and [12].

In order to take into account linguistic variables in k-NN method, it is necessary to define a new function of similarity and new local similarity measures for fuzzy features. Following, we propone several alternatives for substituting the term $sim_i(q_i,c_i)$ in expression (1) for fuzzy features.

In expressions (5), (6), (7), (8), (9) and (10) we use the Principle of Maximum membership degree [17]. For simplicity we assume that the value q_i relatively belongs to the linguistic term α and c_i to β,.

The first local similarity metric states that two feature values are equal if and only if their corresponding linguistic terms are equal.

$$sim_i(q_i,c_i)=\begin{cases}1 & \text{if } \alpha=\beta\\ 0 & \text{in other case}\end{cases} \tag{5}$$

The practical effect of applying this expression is to discretize the domain of the feature i using the cut points of consecutive membership functions as interval limits. The main limitation of this approach is that it supposes those values that are close to interval limits but on different sides as totally different. For those values, more than one membership function with positive image exists.

In order to handle this negative effect we must consider local similarity metrics with image in [0..1]. The following expression is the result.

$$sim_i(q_i,c_i)=\begin{cases}1 & \text{if } \alpha=\beta\\ 1-|\mu_\alpha(q_i)-\mu_\alpha(c_i)| & \text{in other case}\end{cases} \tag{6}$$

That is, dissimilitude between both real values can be measured as the difference between its membership degrees to fuzzy set where first one relatively belong to. The difficulty now consists in the non symmetry of this function. Symmetry is a desired property of similarity metrics. That is why we improve this expression in the next one:

$$sim_i(q_i,c_i)=\begin{cases}1 & \text{if} \quad \alpha=\beta \\ 1-\dfrac{|\mu_\alpha(q_i)-\mu_\alpha(c_i)|+|\mu_\beta(q_i)-\mu_\beta(c_i)|}{2} & \text{in other case}\end{cases} \tag{7}$$

The definition of membership function by itself is a good similarity metric in the sense that it expresses how close the parameter is to the most representative value in the fuzzy set. In the following expression we consider this fact:

$$sim_i(q_i,c_i)=\mu_\beta(c_i) \tag{8}$$

Sometimes it is useful not to consider small values for membership functions. In this case we can use:

$$sim_i(q_i,c_i)=\begin{cases}1 & \text{if} \quad \mu_\beta(c_i)\geq k \\ \mu_\beta(c_i) & \text{in other case}\end{cases} \tag{9}$$

where $k \in (0,1)$, k represents an alpha level (α-cut)_ an alpha level is a threshold restriction on the domain based on the membership degree. α-cut defines a set which contains all the domain values that are part of the fuzzy set at a minimum membership value of α.

Finally, we can compare real values considering the similarity of the corresponding linguistic terms. It is the same as comparing two fuzzy sets.

$$sim_i(q_i,c_i)=\begin{cases}1 & \text{if} \quad \alpha=\beta \\ 1-d(\alpha,\beta) & \text{in other case}\end{cases} \tag{10}$$

The expressions proposed in [18], [17], [19], [20], [21], and [22], can be used to calculate $d(\alpha,\beta)$. For example, expressions (11) and (12):

$$d(\alpha,\beta)=\left(\sum_{i=1}^{n}|\varphi_\alpha(x_i)-\varphi_\beta(x_i)|^r\right)^{1/r} \quad r\geq 1 \tag{11}$$

$$d(\alpha,\beta)=\sup_x|\varphi_\alpha(x)-\varphi_\beta(x)| \tag{12}$$

3 Experimental Results

3.1 Applying a k-NN Approach for Cutting Data Estimation

Process Planning [23, 24] involves setting up a series of detailed instructions to turn a blueprint into a final product. It is the critical link between design and manufacture. The detailed plan contains the route, the processes, the process parameters, the machines and the tools required by production. These instructions determine the cost quality and speed of production. That is why process planning is such a key factor in a production system. However, the complexity of the model does not allow a complete analytical model. For this reason, planning must be based on experience. Besides, an important characteristic of manual process planning is to use examples that have proved successful in similar situations.

Taking into consideration the factors mentioned above, we have built an application prototype that makes use of previous experiences for manufacturing process planning of symmetrical rotatory parts. Next we address the use of previously given expressions for cutting data estimation by retrieving similar cases.

The goal is to estimate the cutting speed for machining a given work piece taking into account several features relative to the piece. Case consists of 9 features that have been identified as useful for this prediction plus a numeric value corresponding to optimal cutting speed.

The solution model consists of parameter estimation with k nearest neighbors (k-NN). Given a query vector q and a objects set CB, the k most similar objects are retrieved to predict the query objective value. The expression (1) is used as similarity function. For nominal features the function of comparison used is expression (2), for numerical features two variants are studied: Equations (2) and (7). The next figure is an example of fuzzy sets defined for the numeric feature hardness.

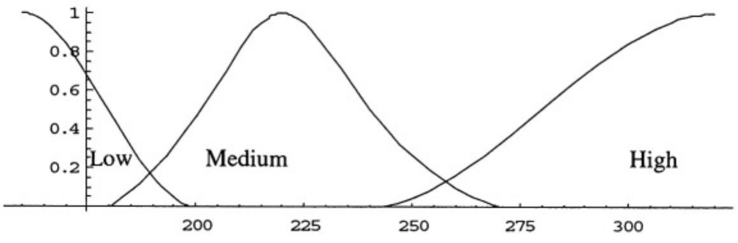


Fig. 2. Fuzzy sets for "hardness" variable

The value predicted for q is taken as the neighbor's objective value average, weighted by its similarity: i.e. expression (4). For measuring the accuracy of a system during learning, one can use a Leave-One-Out-Cross Validation over a given training set [25].

$$LOOCV = \sum_{q \in CB} (q_g - q'_g)^2 \tag{13}$$

Every object from the case base is used as query for estimating objective value (q'_g) using all objects in the case base except itself. The real value for the goals is known. It allows for the estimation of the local error by computing the difference.

The resulting coefficient is calculated from (13). Each feature weight can be learned by using conjugate gradient algorithm. The LOOCV coefficient can be used as the error function. The derivation of this function with respect to the weight takes the form:

$$\frac{\partial E}{\partial w_a} = -2 \cdot \sum_{q \in CB}^{n} \left(q_c - q_c^{'}\right) \cdot \frac{\partial q_c^{'}}{\partial w_a} \tag{14}$$

where:

$$\frac{\partial q_c^{'}}{\partial w_a} = \frac{2 \cdot \sum_{r \in K} \left[\left(q_c - q_c^{'}\right) \cdot \beta(q,r) \cdot \delta_a(q_a, r_a)\right]}{\sum_{r \in K} \beta(q,r)^2} \tag{15}$$

The weight (w_i) in expression (1) was estimated by using the conjugate gradient according to expressions (14) and (15).

The case base is made up of 800 objects taken from workshops specialized in machining these parts. The study was carried out considering two variants based on the proposed system: one (variant I) in which the comparison functions for the numerical features is the expression (2) and another (variant II) in which the fuzzy approach was employed, that is, the expression (7) as a comparison function for these features. The evaluation of the behavior of each variant was carried out using the expression (14). The result of the comparison is shown on table 1.

Table 1. Evaluation results using the LOOCV coefficient

	LOOCV
Variant I	14.146
Variant II	6.213

If we compare these errors by using the Fisher Snedecor test and the quotient in expression (16) the result F= 2.2768. This value is greater than the critical point (1.12342) for m=n=800 and 95% of confidence leading us to conclude that Variant II clearly outperforms Variant I.

$$F = \frac{\frac{LOOCE_1}{m}}{\frac{LOOCE_2}{n}} = \frac{LOOCE_1}{LOOCE_2} \quad \text{for } m = n \tag{16}$$

4 Related Works

A brief description of some existing Fuzzy CBR systems follows. For a more detailed review, see [29].

The memory organization of the ARC system [26] is a hierarchy of classes and cases. Each class is represented by a fuzzy prototype describing the features common to most of the cases belonging to the class. The retrieval step consists of selecting the most promising classes by means of a fuzzy pattern-matching algorithm. Next, cases are selected based on the similarities and differences between the classes and the cases. Finally, near misses are used to avoid repeating past failures.

The CAREFUL system [27] focuses on the first two steps of a case based reasoner: case and problem representation and case retrieval. The representation is based on a hierarchy of fuzzy classes. Fuzzy sets represent imprecise values of the attributes of the cases. The retrieval process proceeds in two steps. First, the problem specification and case filtering, which guides the operator in specifying the problem and identifies potentially interesting cases, and second, the selection that chooses the nearest cases. In the second step, each value of a problem attribute represents a fuzzy constraint and the process selects those cases that better satisfy the constraints according to a weighted fuzzy pattern matching technique.

In PROFIT [28], a fuzzy CBR system was developed to estimate residential property values for real estate transactions. The system enhances CBR techniques with fuzzy predicates expressing preferences in determining similarities between subject and comparable properties. These similarities guide the selection and aggregation process, leading to the final property value estimate. PROFIT has been successfully tested on thousands of real estate transactions.

5 Conclusions

When we need to compare objects, the use of fuzzy features is suggested for several reasons. We state some in this paper. Also several local similarity measures are presented for this purpose.

The similarity functions obtained are useful in different computational models such as Pattern recognition techniques, Case-based reasoning and Machine learning, particularly in learning, from examples [29]. The use of linguistic variables to represent the features made it possible to reduce the number of necessary examples to learn a class or a concept. The functions proposed in this work can be used to measure the similarity between the examples and the representation of the concept or class that is built.

In particular, some of the functions proposed in this paper are used to implement a retrieving procedure in k-NN procedures in applications of estimation of functions, and in the planning process described in epigraphic 3. The experimental results are successful.

References

1. Zadeh, L.A.. Fuzzy Sets. Information and Control, 8, pp. 338-353. 1965.
2. Herrmann, C.S.. Fuzzy logic as interfacing techniques in hybrid AI-systems. In Fuzzy Logic in Artificial Intelligence, IJCAI Workshop, Springer LNAI, 1995.
3. Chmielewski, M.R. and Grzymala-Busse, JW.. Global discretization of continuous attributes as preprocessing for machine learning. International Journal of Approximate Reasoning, 15:319-331. 1996.
4. Catlett, J.. On changing continuous attributes into ordered discrete attributes. In Kodratoff, Y., ed., Proceeding of the European Working Session on Learning, 164-178, 1991, Springer-Verlag.

5. Garcia, MM. and Bello, R.. A model and its different applications to case-based reasoning. Knowledge-based systems 9, 465-473. 1996.
6. Baldwin, J.F. and Ribeiro, R.A.. Fuzzy reasoning by case for decision support systems. International Journal of Uncertainty, Fuzziness and Knowledge-based systems, vol. 2, no. l, pp. 11-24. 1991.
7. Mechitov, AI. et al, An ordinal model for case-based reasoning in a classification task. International Journal of Expert Systems, vol. 9, no. 2, pp. 225-242, 1996.
8. Dubois, D. et al.. Weighted fuzzy pattern matching. Fuzzy sets and systems, 28, 351-362, 1988.
9. Plaza, E. et al.. A logical approach to case-based reasoning using fuzzy similariy relations. Information Sciences. 1997.
10. Jaczynski, M. and Trousse, B.. Fuzzy logic for the retrieval step of a Case-based reasoner. EWCBR-94, 7-10th of November, France, 313-322. 1994.
11. Petersen, J.. Similarity of fuzzy data in a case-based fuzzy systems in anaesthes. Fuzzy Sets and Systems 85, pp. 247-262. 1997.
12. Lopez de Mantaras, R. and Godo, L..From intervals to fuzzy truth-values: adding flexibility to reasoning under uncertainty. International Journal of Uncertainty, Fuzziness and KBS vol. 5, no. 3, pp. 251-260. 1997.
13. Bonissone, PP. and Ayud, S.. Similarity measures fort case-based reasoning systems. In proceedings of the International Conference on Information Processing and Management of Uncertainty in KBS (IPMU 92). Mallorca, July 6-10, 483-487. 1992.
14. Cox, E.. The fuzzy systems handbook. AP Professional, 1994.
15. Jang, J.R. and Sun, Ch.T.. Neuro-fuzzy modeling and control. Proceeding of the IEEE, march 1995.
16. Jang, J.R., et al.. Neuro-fuzzy and soft computing: a computational approach to learning and machine intelligence. Prentice Hall, 1997.
17. Shi-quan, C.. Fuzzy equivalence and multiobjective decision making. In Fuzzy Computing, ed. M.M. Gupta and T. Yamakawa, Elsevier Science Publishers (North-Holland), 1988.
18. Zwick, R., Carlstein, E. and Budescu, D.. Measures of similarity among fuzzy concepts: a comparative analysis. International Journal of Approximate reasoning, 1987.
19. Bortolan, G. and Degani, R.. Linguistic approximation of fuzzy certainty factors in computerized electrocardiography. In Fuzzy Computing, ed. by M.M. Gupta and T. Yamakawa, Elsevier Science Publishers (North-Holland), 1988.
20. Wang, W.. New similarity measures on fuzzy sets and on elements. Fuzzy Sets and Systems 85, pp. 305-309. 1997.
21. Fan, J. and Xie, W.. Some notes on similarity measure and proximity measure. Fuzzy Sets and Systems 101, pp. 403-412. 1999.
22. Fan, J. and Xie, W.. Distance measure and induced fuzzy entropy. Fuzzy Sets and Systems 104, 305-314. 1999.
23. McAllester, D. and Rosenblitt, D.. Systematic nonlinear planning. In Proceedings of AAAI-91, pages 634-639. 1991.
24. Nau, D.S., Gupta, SK. and W.C. Regli, WC.. AI planning verses manufacturing-operation planning: A case study. In Proceedings of the International Joint Conference on Artificial Intelligence, pp. 1670--1676, August 1999.
25. Kohavi, R., "A study of cross-validation and bootstrap for accuracy estimation and model selection," International Joint Conference on Artificial Intelligence (IJCAI). 1995.
26. Bonissone, P. and Lopez de Mantaras, R. 1998. Fuzzy Case-based Reasoning Systems, *Handbook on Fuzzy Computing* (F 4.3), Oxford University Press

27. Plaza, E. and Lopez de Mantaras, R. 1990. A Case-Based Apprentice that Learns from Fuzzy Examples, Methodologies for Intelligent Systems, 5th edition, Ras, Zemankova and Emrich, Elsevier, pp 420-427
28. Jaczynski, M. and Trousse, B. 1994. Fuzzy Logic for the Retrieval Step of a Case-Based Reasoner, *Proceedings Second European Workshop on Case-Based Reasoning,* pp 313-322.
29. Bonissone, P. and Cheetham, W. 1998. Fuzzy Case-Based Reasoning for Residential Property Valuation, *Handbook on Fuzzy Computing* (G 15.1), Oxford University Press

Decomposing Ordinal Sums in Neural Multi-adjoint Logic Programs*

Jesús Medina, Enrique Mérida-Casermeiro, and Manuel Ojeda-Aciego

Dept. Matemática Aplicada, Universidad de Málaga
{jmedina, merida, aciego}@ctima.uma.es

Abstract. The theory of multi-adjoint logic programs has been introduced as a unifying framework to deal with uncertainty, imprecise data or incomplete information. From the applicative part, a neural net based implementation of homogeneous propositional multi-adjoint logic programming on the unit interval has been presented elsewhere, but restricted to the case in which the only connectives involved in the program were the usual product, Gödel and **Łukasiewicz** together with weighted sums.

A modification of the neural implementation is presented here in order to deal with a more general family of adjoint pairs, including conjunctors constructed as an ordinal sum of a finite family of basic conjunctors. This enhancement greatly expands the scope of the initial approach, since every t-norm (the type of conjunctor generally used in applications) can be expressed as an ordinal sum of product, Gödel and **Łukasiewicz** conjunctors.

1 Introduction

The study of reasoning methods under uncertainty, imprecise data or incomplete information has received increasing attention in the recent years. A number of different approaches have been proposed with the aim of better explaining observed facts, specifying statements, reasoning and/or executing programs under some type of uncertainty whatever it might be.

One important and powerful mathematical tool that has been used for this purpose at theoretical level is fuzzy logic. From the applicative side, neural networks have a massively parallel architecture-based dynamics inspired by the structure of human brain, adaptation capabilities, and fault tolerance.

Using neural networks in the context of logic programming is not a completely novel idea; for instance, in [1] it is shown how fuzzy logic programs can be transformed into neural nets, where adaptations of uncertainties in the knowledge base increase the reliability of the program and are carried out automatically.

Regarding the approximation of the semantics of logic programs, the fixpoint of the $T_{\mathbb{P}}$ operator for a certain class of classical propositional logic programs (called acyclic logic programs) is constructed in [2] by using a 3-layered recurrent neural network, as a means of providing a massively parallel computational model for logic programming; this result is later extended in [3] to deal with the first order case.

* Partially supported by project TIC2003-09001-C02-01.

Recently, a new approach presented in [5] introduced a hybrid framework to handling uncertainty, expressed in the language of multi-adjoint logic but implemented by using ideas from the world of neural networks. This neural-like implementation of multi-adjoint logic programming was presented with the restriction that the only connectives involved in the program were the usual product, Gödel and Łukasiewicz together with weighted sums. Since the theoretical development of the multi-adjoint framework does not rely on particular properties of the product, Gödel and Łukasiewicz adjoint pairs, it seems convenient to allow for a generalization of the implementation to admit, at least, a family of continuous t-norms (recall that any continuous t-norm can be interpreted as the ordinal sum of product and Łukasiewicz t-norms).

The purpose of this paper is to present a refined version of the neural implementation, such that conjunctors which are built as ordinal sums of product, Gödel and Łukasiewicz t-norms are decomposed into its components and, as a result, the original neural approach is still applicable. It is worth to remark that the learning capabilities of neural networks are not used in the implementation; this should not be considered a negative feature, because it is precisely in the final goal of this research line, a neural approach to abductive multi-adjoint logic programming, when learning will play a crucial role.

The structure of the paper is as follows: In Section 2, the syntax and semantics of multi-adjoint logic programs are introduced; in Section 3, the new proposed neural model for homogeneous multi-adjoint programs is presented in order to cope with conjunctors defined as ordinal sums, a high level implementation is introduced and proven to be sound. The paper finishes with some conclusions and pointers to future work.

2 Preliminary Definitions

To make this paper as self-contained as possible, the necessary definitions about multi-adjoint structures are included in this section. For motivating comments, the interested reader is referred to [6].

Multi-adjoint logic programming is a general theory of logic programming which allows the simultaneous use of different implications in the rules and rather general connectives in the bodies.

The first interesting feature of multi-adjoint logic programs is that a number of different implications are allowed in the bodies of the rules. The basic definition is the generalization of residuated lattice given below:

Definition 1. *A* multi-adjoint lattice *$\mathcal{L}$ is a tuple $(L, \preceq, \leftarrow_1, \&_1, \dots, \leftarrow_n, \&_n)$ satisfying the following items:*

1. *$\langle L, \preceq\rangle$ is a bounded lattice, i.e. it has bottom and top elements;*
2. *$\top \&_i \vartheta = \vartheta \&_i \top = \vartheta$ for all $\vartheta \in L$ for $i = 1, \dots, n$;*
3. *$(\&_i, \leftarrow_i)$ is an* adjoint pair *in $\langle L, \preceq\rangle$ for $i = 1, \dots, n$; i.e.*
   (a) Operation $\&_i$ is increasing in both arguments,
   (b) Operation $\leftarrow_i$ is increasing in the first argument and decreasing in the second,
   (c) For any $x, y, z \in P$, we have $x \preceq (y \leftarrow_i z)$ if and only if $(x \&_i z) \preceq y$.

In the rest of the paper we restrict to the unit interval, although the general framework of multi-adjoint logic programming is applicable to a general lattice.

2.1 Syntax and Semantics

A *multi-adjoint program* is a set of weighted rules $\langle F, \vartheta \rangle$ satisfying

1. F is a formula of the form $A \leftarrow_i \mathcal{B}$ where A is a propositional symbol called the *head* of the rule, and $\mathcal{B}$ is a well-formed formula, which is called the *body,* built from propositional symbols $B_1, \dots, B_n$ $(n \geq 0)$ by the use of monotone operators.
2. The *weight* ϑ is an element (a truth-value) of [0,1].

Facts are rules with body 1 and a *query* (or *goal*) is a propositional symbol intended as a question? A prompting the system.

Once presented the syntax of multi-adjoint programs, the semantics is given below.

Definition 2. *An* interpretation *is a mapping I from the set of propositional symbols Π to the lattice $\langle [0,1], \leq \rangle$.*

Note that each of these interpretations can be uniquely extended to the whole set of formulas, and this extension is denoted as $\hat{I}$. The set of all the interpretations is denoted $\mathcal{I}_{\mathfrak{L}}$.

The ordering $\leq$ of the truth-values L can be easily extended to $\mathcal{I}_{\mathfrak{L}}$, which also inherits the structure of complete lattice and is denoted $\sqsubseteq$. The minimum element of the lattice $\mathcal{I}_{\mathfrak{L}}$, which assigns 0 to any propositional symbol, will be denoted Δ.

Definition 3.

1. *An interpretation $I \in \mathcal{I}_{\mathfrak{L}}$* satisfies *$\langle A \leftarrow_i \mathcal{B}, \vartheta \rangle$ if and only if $\vartheta \leq \hat{I}\,(A \leftarrow_i \mathcal{B})$.*
2. *An interpretation $I \in \mathcal{I}_{\mathfrak{L}}$ is a* model *of a multi-adjoint logic program $\mathbb{P}$ iff all weighted rules in $\mathbb{P}$ are satisfied by I.*
3. *An element $\lambda \in L$ is a* correct answer *for a program $\mathbb{P}$ and a query ? A if for any interpretation $I \in \mathcal{I}_{\mathfrak{L}}$ which is a model of $\mathbb{P}$ we have $\lambda \leq I(A)$.*

The operational approach to multi-adjoint logic programs used in this paper will be based on the fixpoint semantics provided by the immediate consequences operator, given in the classical case by van Emden and Kowalski [7], which can be generalised to the multi-adjoint framework by means of the adjoint property, as shown below:

Definition 4. *Let $\mathbb{P}$ be a multi-adjoint program; the* immediate consequences operator, *$T_{\mathbb{P}}\colon \mathcal{I}_{\mathfrak{L}} \to \mathcal{I}_{\mathfrak{L}}$, maps interpretations to interpretations, and for $I \in \mathcal{I}_{\mathfrak{L}}$ and $A \in \Pi$ is given by*

$$T_{\mathbb{P}}(I)(A) = \sup\left\{\vartheta \,\&_i\, \hat{I}(\mathcal{B}) \mid \langle A \leftarrow_i \mathcal{B}, \vartheta \rangle \in \mathbb{P}\right\}$$

As usual, it is possible to characterise the semantics of a multi-adjoint logic program by the post-fixpoints of $T_{\mathbb{P}}$; that is, an interpretation I is a model of a multi-adjoint logic program $\mathbb{P}$ iff $T_{\mathbb{P}}(I) \sqsubseteq I$. The $T_{\mathbb{P}}$ operator is proved to be monotonic and continuous under very general hypotheses.

Once one knows that $T_{\mathbb{P}}$ can be continuous under very general hypotheses [6], then the least model can be reached in at most countably many iterations beginning with the least interpretation, that is, the least model is $T_{\mathbb{P}} \uparrow \omega(\Delta)$.

2.2 Homogeneous Programs

Regarding the implementation as a neural network of [5], the introduction of the so-called *homogeneous rules* provided a simpler and standard representation for any multi-adjoint program.

Definition 5. *A weighted formula is said to be* homogeneous *if it has one of the forms:*

- $\langle A \leftarrow_i \&_i(B_1,\dots,B_n),\vartheta\rangle$
- $\langle A \leftarrow_i @(B_1,\dots,B_n),1\rangle$
- $\langle A \leftarrow_i B_1,\vartheta\rangle$

where $A, B_1,\dots,B_n$ are propositional symbols, $(\&_i,\leftarrow_i)$ is an adjoint pair, and $@$ is an aggregator.

The homogeneous rules represent exactly the simplest type of (proper) rules one can have in a program. The way in which the translation of a general multi-adjoint program is homogenized is irrelevant for the purposes of this paper; anyway, it is worth mentioning that it is a model preserving procedure with linear complexity.

For the sake of self-contention, a brief overview of the neural network from [5] is presented below.

2.3 The Restricted Neural Implementation

A neural network was considered in which each process unit is associated either to a propositional symbol of the initial program or to an homogeneous rule of the transformed program. The state of the i-th neuron in the instant t is expressed by its output function, denoted $S_i(t)$. The state of the network can be expressed by means of a state vector $S(t)$, whose components are the output of the neurons forming the network; the initial state of S is 0 for all the components except those representing a propositional variable, say A, in which case its value is defined to be:

$$S_A(0) = \begin{cases} \vartheta_A & \text{if } \langle A \leftarrow 1,\vartheta_A\rangle \in \mathbb{P}, \\ 0 & \text{otherwise.} \end{cases}$$

where $S_A(0)$ denotes the component associated to a propositional symbol A.

The connection between neurons is denoted by a matrix of weights W, in which w_{kj} indicates the existence or absence of connection between unit k and unit j; if the neuron represents a weighted sum, then the matrix of weights also represents the weights associated to any of the inputs. The weights of the connections related to neuron i (i.e., the i-th row of W) are represented by a vector $w_{i\bullet}$, and are allocated in an internal vector register of the neuron, which can be seen as a distributed information system.

The initial truth-value of the propositional symbol or homogeneous rule v_i is loaded in the internal register, together with a signal m_i to distinguish whether the neuron is associated either to a fact or to a rule; in the latter case, information about the type of operator is also included. Therefore, there are two vectors: one storing the truth-values v of atoms and homogeneous rules, and another m storing the type of the neurons.

The signal m_i indicates the functioning mode of the neuron. If $m_i = 1$, then the neuron is assumed to be associated to a propositional symbol, and its next state is the maximum value among all the operators involved in its input, its previous state, and the initial truth-value v_i. More precisely:

$$S_i(t+1) = \max\left\{v_i, \max_k\{S_k(t) \mid w_{ik} > 0\}\right\}.$$

When a neuron is associated to the product, Gödel, or Łukasiewicz implication, respectively, then the signal m_i is set to 2, 3, and 4, respectively. Its input is formed by the external value v_i of the rule, and the outputs of the neurons associated to the body of the implication.

The output of the neuron mimics the behaviour of the implication in terms of the adjoint property when a rule of type m_i has been used; specifically, the output in the next instant will be:

$$S_i(t+1) = \begin{cases} v_i \prod\limits_{k \mid w_{ik}>0} S_k(t) & \text{if } m_i = 2 \\ \min\left\{v_i, \min\limits_k\{S_k(t) \mid w_{ik} > 0\}\right\} & \text{if } m_i = 3 \\ \max\{v_i + \sum\limits_{k \mid w_{ik}>0} (S_k(t) - 1), 0\} & \text{if } m_i = 4 \end{cases}$$

A neuron associated to a weighted sum has signal $m_i = 5$, and its output is

$$S_i(t+1) = \sum_k w'_{ik} S_k(t) \qquad \text{where} \qquad w'_{ik} = \frac{w_{ik}}{\sum\limits_r w_{ir}}$$

3 Towards a New Model of Generic Neuron

As stated above, the neural net implementation of the immediate consequences operator of an homogeneous program was introduced for the case of the multi-adjoint lattice $([0,1], \leq, \&_P, \leftarrow_P, \&_G, \leftarrow_G, \&_L, \leftarrow_L)$. As the theoretical development of the multi-adjoint framework does not rely on particular properties of these three adjoint pairs, it seems convenient to allow for a generalization of the implementation to, at least, a family of continuous t-norms, since any continuous t-norm can be interpreted as the ordinal sum of product and Łukasiewicz t-norms.

Recall the definition of ordinal sum of a family of t-norms:

Definition 6. *Let $(\&_i)_{i \in A}$ be a family of t-norms and a family of non-empty pairwise disjoint subintervals $[a_i, b_i]$ of* [0,1]. *The* ordinal sum *of the summands $(a_i, b_i, \&_i)$, $i \in A$ is the t-norm & defined as*

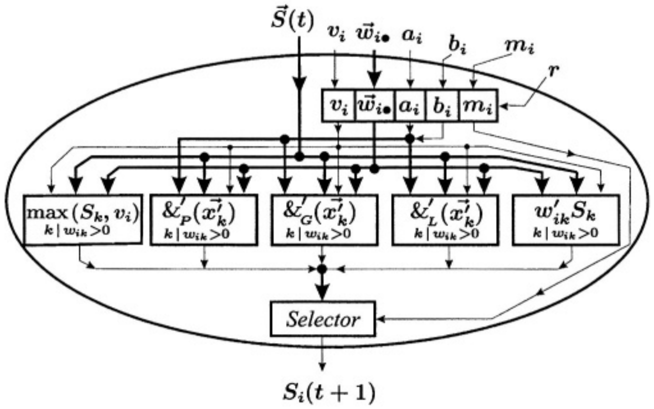
$$\&(x,y) = \begin{cases} a_i + (b_i - a_i)\,\&_i\left(\frac{x-a_i}{b_i-a_i}, \frac{y-a_i}{b_i-a_i}\right) & \textit{if } x, y \in [a_i, b_i) \\ \min(x,y) & \textit{otherwise} \end{cases}$$

3.1 The Proposed Model of Generic Neuron

The model of neuron presented in [5] has to be modified in order to be able to represent conjunctors defined as ordinal sums of product and Łukasiewicz conjunctions.

To begin with, two new registers are introduced so that it is possible to represent the (sub-)interval on which the connective will be operating, i.e. to set the values of the points a_i and b_i.

The generic neuron shown below has a binary signal r, which is shared by all the neurons. If $r = 1$, then it is possible to modify the content of the internal registers v, $w_{i.}$, m_i, a_i and b_i by means of the external signals of the same name. If $r = 0$, then neuron computes an output $S_i(t+1)$ in terms of its own internal registers and its network inputs (that is, the outputs of the rest of neurons in the previous step $S(t)$).


$\vec{S}(t)$
v_i $\vec{w}_{i\bullet}$ a_i b_i m_i
r
v_i $\vec{w}_{i\bullet}$ a_i b_i m_i
$\max(S_k, v_i)$ $k \mid w_{ik}>0$
$\&'_P(\vec{x}'_k)$ $k \mid w_{ik}>0$
$\&'_G(\vec{x}'_k)$ $k \mid w_{ik}>0$
$\&'_L(\vec{x}'_k)$ $k \mid w_{ik}>0$
$w'_{ik}S_k$ $k \mid w_{ik}>0$
Selector
$S_i(t+1)$

Depending on the content of the register m_i, the selector generates the component of the generic neuron which gets activated. When $m_i = 1$ the value of the registers a_i and b_i is irrelevant; this is also true when $m_i = 5$. In this case the neuron operates as a weighted sum which outputs

$$S_i(t+1) = \sum_{k \mid w_{ik}>0} w'_{ik} S_k(t) \quad \text{where} \quad w'_{ik} = \frac{w_{ik}}{\sum\limits_{k \mid w_{ik}>0} w_{ik}}$$

In the cases $m_i = 2$, $m_i = 3$ and $m_i = 4$ the neuron uses the input $S(t)$ and the value v_i to compute a vector $x_i{}'$ as follows:

- Only the value v_i and the components k of $S(t)$ such that $w_{ik} > 0$ are considered.
- The vector x' is computed as $x'_k = \begin{cases} 1 & S_k(t) \geq b_i \\ \frac{S_k(t)-a_i}{b_i-a_i} & a_i \leq S_k(t) < b_i \\ 0 & S_k(t) < a_i \end{cases}$
- The output of the neuron is[1] $S_i(t+1) = \begin{cases} a_i + (b_i - a_i)\&_P(x') & \text{if } m_i = 2 \\ a_i + (b_i - a_i)\&_G(x') & \text{if } m_i = 3 \\ a_i + (b_i - a_i)\&_L(x') & \text{if } m_i = 4 \end{cases}$

[1] The functions corresponding to each case are represented in the neuron as $\&'_P$, $\&'_G$, $\&'_L$.

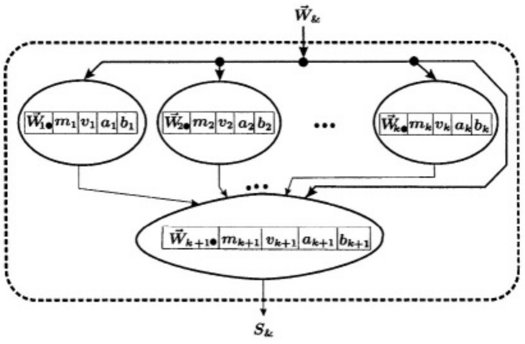
3.2 Neural Representation of Ordinal Sums

Consider a conjunctor represented as an ordinal sum

$$\& = \{\langle a_1, b_1, \&_1\rangle, \langle a_2, b_2, \&_2\rangle, \dots, \langle a_k, b_k, \&_k\rangle\}$$

where $\&_i$ can be either $\&_P$ or $\&_L$ (since Gödel conjunctor operates by default out of the intervals $[a_i, b_i)$).

The implementation of such a compound connective needs $k+1$ neurons,thefirst k are associated to each subinterval of the ordinal sum, and the final one collects all their outputs and generates the final output of the sum. As shown in the picture below, all the neurons of the block associated with the ordinal sum & receive the output of the neurons associated to the propositions in the body of the rule, whereas the collecting neuron receives the output of the rest of the neurons integrating the block of the ordinal sum.


$\vec{W}_\&$
$\vec{W}_{1\bullet}$ m_1 v_1 a_1 b_1
$\vec{W}_{2\bullet}$ m_2 v_2 a_2 b_2
$\vec{W}_{k\bullet}$ m_k v_k a_k b_k
$\vec{W}_{k+1\bullet}$ m_{k+1} v_{k+1} a_{k+1} b_{k+1}
$S_\&$

Let us see the process working on a specific toy example:

Example 1. Consider the ordinal sum defined by $\& = \{(0.1, 0.5, \&_P), (0.7, 0.9, \&_L)\}$ and the program with one rule and two facts: $\langle p \leftarrow_\& q \,\&\, s, 0.6\rangle, \langle q, 0.4\rangle, \langle s, 0.3\rangle$. The neural representation needs 6 neurons: three are associated to the propositions p, q and s, and the other three are used to implement the rule.

The initialization of the registers allocates the values for the vectors $\boldsymbol{m}, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{v}$ and for the weights matrix as shown below

$$\begin{aligned} \boldsymbol{m} &= (1, 1, 1, 2, 4, 3) \\ \boldsymbol{a} &= (0.0, 0.0, 0.0, 0.1, 0.7, 0.0) \\ \boldsymbol{b} &= (1.0, 1.0, 1.0, 0.5, 0.9, 1.0) \\ \boldsymbol{v} &= (0.0, 0.4, 0.3, 0.6, 0.6, 0.6) \end{aligned} \qquad \boldsymbol{W} = \begin{pmatrix} 0&0&0&0&0&1 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&1&1&0&0&0 \\ 0&1&1&0&0&0 \\ 0&1&1&1&1&0 \end{pmatrix}$$

3.3 Implementation

A number of simulations have been obtained through a MATLAB implementation in a conventional sequential computer. A high level description of the implementation is given below:

1. **Initialize** the network with the appropriate values of $\boldsymbol{v}$, $\boldsymbol{m}$, W and, in addition, a tolerance value *tol* to be used as a stop criterion. The output $S_i(t)$ of the neurons associated to facts (which are propositional variables, so $m_i = 1$) are initialized with its truth-value v_i.
2. **Repeat**
   Update all the states S_i of the neurons of the network:
   (a) If $m_i = 1$, then:
      i. Construct the set $J_i = \{j \mid w_{ij} = 1\}$. In this case, this amounts to collect all the rules with head A.
      ii. Then, update the state of neuron i as follows:

$$S_i(t) = \begin{cases} \max\{v_i, \max_{J_i} S_j(t-1)\} & \text{if } J_i \neq \varnothing \\ v_i & \text{otherwise} \end{cases}$$

   (b) If $m_i = 2, 3, 4$, then:
      i. Find the neurons j (if any) which operate on the neuron i, that is, construct the set $J_i = \{j \mid w_{ij} = 1\}$.
      ii. Then, update the state of neuron i as follows:

$$S_i(t) = \begin{cases} v_i \prod_{J_i} S_j(t-1) & \text{if } m_i = 2 \\ \min\{v_i, \min_{J_i} S_j(t-1)\} & \text{if } m_i = 3 \\ \max\{v_i + \sum_{J_i}(S_j(t-1) - 1), 0\} & \text{if } m_i = 4 \end{cases}$$

   Recall that when $m_i = 2, 3, 4$ the neuron corresponds to a product, Gödel, Łukasiewicz (resp.) implication.

   (c) If $m_i = 5$, then the neuron corresponds to an aggregator, and its update follows a different pattern:
      i. Determine the set $K_i = \{j \mid w_{ij} > 0\}$ and calculate $sum = \sum_{K_i} w_{ij}$
      ii. Update the neuron as follows:

$$S_i(t) = \frac{1}{sum} \sum_{K_i} w_{ij} \cdot S_j(t-1)$$

   Until the stop criterion $\|\boldsymbol{S}(t) - \boldsymbol{S}(t-1)\| < tol$ is fulfilled, where $\|\cdot\|$ denotes euclidean distance.

Example 2. Consider a program with facts $\langle p \leftarrow 1, 0.3\rangle$ and $\langle q \leftarrow 1, 0.4\rangle$ and three rules $\langle p \leftarrow_{\&} s \,\&\, q, 0.8\rangle, \langle s \leftarrow_{\&} q, 0.7\rangle, \langle p \leftarrow_{L} s, 0.5\rangle$ where the conjunction & is defined as an ordinal sum by $\& = \{(0.1, 0.5, \&_P), (0.7, 0.9, \&_L)\}$.

The net for the program will consist of ten neurons, and is sketched in Fig. 1, three of which represent variables p, q, s, and the rest are needed to represent the rules (three for each ordinal sum rule and another one for the Łukasiewicz rule).

$$\begin{aligned} \boldsymbol{m} &= (1,1,1,2,4,3,2,4,3,4) \\ \boldsymbol{a} &= (0.0,0.0,0.0,0.1,0.7,0.0,0.1,0.7,0.0,0.0) \\ \boldsymbol{b} &= (1.0,1.0,1.0,0.5,0.9,1.0,0.5,0.9,1.0,1.0) \\ \boldsymbol{v} &= (0.3,0.4,0.0,0.8,0.8,0.8,0.7,0.7,0.7,0.5) \end{aligned} \qquad \boldsymbol{W} = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & 1 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot \\ \cdot & 1 & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & 1 & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & 1 & 1 & 1 & 1 & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & 1 & \cdot & \cdot & \cdot & \cdot & 1 & 1 & \cdot & \cdot \\ \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix}$$

After running the net, its state vector gets stabilized at

$$\boldsymbol{S} = (0.325, 0.4, 0.4, 0.325, 0.7, 0.325, 0.4, 0.7, 0.4, 0)$$

where the last seven components correspond to hidden neurons, the first ones are interpreted as the obtained truth-value for p, q and s.

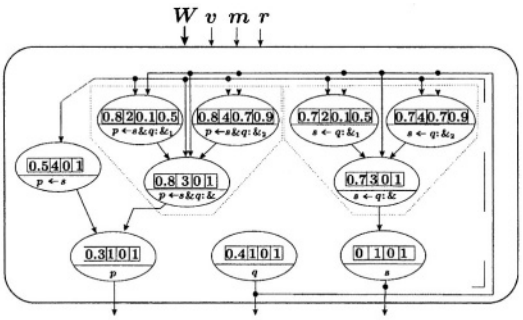


Fig. 1. A network for Example 2

3.4 Relating the Net and $T_{\mathbb{P}}$

In this section we relate the behavior of the components of the state vector with the immediate consequence operator. To begin with, it is convenient to recall that the functions S_i implemented by each neuron are non-decresing. The proof is straightforward, after analysing the different cases arising from the type of neuron (i.e. the register m_i).

Regarding the soundness of the implementation sketched above, the following theorem can be obtained, although space restrictions do not allow to include the proof.

Theorem 1. *Given a homogeneousprogram* $\mathbb{P}$ *and a propositional symbol A, then the sequence* $S_A(n)$ *approximates the value of the least model of* $\mathbb{P}$ *in A up to any prescribed level of precision.*

4 Conclusions and Future Work

A new neural-like model has been proposed which extends that recently given to multi-adjoint logic programming. The model mimics the consequences operator in order to obtain the least fixpoint of the immediate consequences operator for a given multiadjoint logic program. This way, it is possible to obtain the computed truth-values of all propositional symbols involved in the program in a parallel way. This extended approach considers, in addition to the three most important adjoint pairs in the unit interval (product, Gödel, and Łukasiewicz) and weighted sums, the combinations as finite ordinal sums of the previous conjunctors.

We have decided to extend the original generic model of neuron, capable of adapting to perform different functions according with its inputs. Although it is possible to consider simpler units, the price to pay is to consider a set of *different units* each type representing a different type of homogeneous rule or proposition. This way one has both advantages and disadvantages: the former are related to the easier description of the units, whereas the latter arise from the greater complexity of the resulting network. A complete analysis of the compromise between simplicity of the units and complexity of the network will be the subject of future work.

Another line of future research deals with further developing the neural approach to abductive multi-adjoint logic programming [4]; it is in this respect where the learning capabilities of neural networks are proving to be an important tool.

References

1. P. Eklund and F. Klawonn. Neural fuzzy logic programming. *IEEE Tr. on Neural Networks,* 3(5):815–818, 1992.
2. S. Hölldobler and Y. Kalinke. Towards a new massively parallel computational model for logic programming. In *ECAI'94 workshop on Combining Symbolic and Connectionist Processing,* pages 68–77,1994.
3. S. Hölldobler, Y. Kalinke, and H.-P. Störr. Approximating the semantics of logic programs by recurrent neural networks. *Applied Intelligence,* 11(1):45–58, 1999.
4. J. Medina, E. Mérida-Casermeiro, and M. Ojeda-Aciego. A neural approach to abductive multi-adjoint reasoning. In *AI - Methodologies, Systems, Applications. AIMSA '02,* pages 213–222, Lect. Notes in Computer Science 2443, 2002.
5. J. Medina, E. Mérida-Casermeiro, and M. Ojeda-Aciego. A neural implementation of multi-adjoint logic programming. *Journal of Applied Logic* 2/3:301–324, 2004.
6. J. Medina, M. Ojeda-Aciego, and P. Vojtáš. Multi-adjoint logic programming with continuous semantics. In *Logic Programming and Non-Monotonic Reasoning, LPNMR'01,* pages 351–364. Lect. Notes in Artificial Intelligence 2173, 2001.
7. M. H. van Emden and R. Kowalski. The semantics of predicate logic as a programming language. *Journal of the ACM,* 23(4):733–742, 1976.

Comparing Metrics in Fuzzy Clustering for Symbolic Data on SODAS Format

Alzennyr Silva, Francisco Carvalho, Teresa Ludermir, and Nicomedes Cavalcanti

Centro de Informática - CIn / UFPE, Av. Prof. Luiz Freire, s/n - Cidade Universitária, CEP 52011-030 Recife - PE, Brazil
{acgs2, fatc, tbl, nlcj}@cin.ufpe.br
http://www.cin.ufpe.br

Abstract. A number of approaches to solve the problem of data clustering are available in the literature. This paper introduces a comparative study in some distances metrics very known in the literature of symbolic data. This work uses an adaptation of an algorithm that applies concepts of fuzzy clustering and then creates groups of individuals characterized by symbolic variables of mixed types. The core of the algorithm consists of a dissimilarity function that can be replaced without collateral effects. The input data is provided on the format of a SODAS file. The results of the experiments on representative databases show the performance of each analyzed metric.

1 Introduction

Cluster analysis is an exploratory data analysis tool whose aim is to organize a set of items (usually represented as a vector of quantitative values in a multidimensional space) into clusters such that items within a given cluster have a high degree of similarity, whereas items belonging to different clusters have a high degree of dissimilarity. Cluster analysis techniques can be divided into hierarchical and partitional methods.

Hierarchical methods yield complete hierarchy, i.e., a nested sequence of input data partitions [17] [18]. Hierarchical methods can be either agglomerative or divisive. Agglomerative methods yield a sequence of nested partitions starting with singletons, where each item is in a unique cluster, and ending with the trivial clustering, where all items are in the same cluster. A divisive method starts with all items in a single cluster and performs divisions until a stopping criterion is met (usually, until obtaining a partition of singleton clusters).

Partitional methods aim to place a single partition of the input data into a fixed number of clusters. These methods identify the partition that optimizes an adequacy criterion. To improve clustering quality, the algorithm runs multiple times using different starting points and the best configuration is selected as the output clustering. Dynamic cluster algorithms are iterative two-step relocation algorithms involving the construction of clusters and the identification of a suitable representation or prototype for each cluster by locally optimizing an adequacy criterion between the clusters and their corresponding representation [7]. The principal feature of this kind of algorithm is the allocation step. This step assigns individuals to the correct class according to their proximity to its prototype. This is followed by a representation step where prototypes are updated according to the assignment of the individuals in the allocation step until convergence of the algorithm is achieved, i.e., the adequacy criterion reaches a stationary value.

The main advantage of fuzzy clustering is: membership values of each individual in the clusters make it possible to evaluate and analyze the results in a more soft way. In this sense, a single individual can belong to more than one cluster, revealing aspects of similarity between this one and all the classes. Hence, it is possible to identify the classes were one particular individual presents greater similarity without discarding similarity between it and further classes. That would not be possible to do in traditional hard clustering.

2 Symbolic Data Analysis

Objects to be clustered are usually represented as a vector of quantitative measurements, but due to the recent advances in database technologies it is now common to record multivalued and interval data.

Symbolic data analysis (SDA) is a new domain in the area of knowledge discovery and data management, related to multivariate analysis, pattern recognition and artificial intelligence. It aims to provide suitable methods (clustering, factorial techniques, decision tree, etc.) for managing aggregated data described through multi-valued variables, where there are sets of categories, intervals, or weight (probability) distributions in the cells of the data table (for more details about SDA, see www.jsda.unina2.it). In this so-called symbolic data table, the rows are the symbolic objects and the columns are the symbolic variables [3].

SDA has provided suitable tools for clustering symbolic data. Concerning hierarchical methods, an agglomerative approach has been introduced which forms composite symbolic objects using a join operator whenever mutual pairs of symbolic objects are selected for agglomeration based on minimum dissimilarity [10] or maximum similarity [11].

In [12] and [13] are presented, respectively, divisive and agglomerative algorithms for symbolic data based on the combined usage of similarity and dissimilarity measures. These proximity (similarity or dissimilarity) measures are defined on the basis of the position, span and content of symbolic objects. In [14] is presented a hierarchical clustering algorithm for symbolic objects based on the gravitational approach, which is inspired on the movement of particles in space due to their mutual gravitational attraction. In [15] is presented an ISODATA clustering procedure for symbolic objects using distributed genetic algorithms.

SDA has also provided partitioning cluster algorithms for symbolic data. In [6] a transfer algorithm is used to partition a set of symbolic objects into clusters described by weight distribution vectors. In [8] it was presented a fuzzy k-means algorithm to cluster symbolic data described by different types of symbolic variables. In [9] it was presented an iterative relocation algorithm to partition a set of symbolic objects into classes so as to minimize the sum of the description potentials of the classes.

The aim of this paper is to design an algorithm that generates fuzzy clusters of symbolic data using distinct functions as dissimilarity metrics. For this, a symbolic data clustering method based on the FSCM (Fuzzy Symbolic C-Means Algorithm) [8] has been implemented. The core of the algorithm consists of a dissimilarity measure that can be easily replaced.

The principal purpose of this paper is to provide a good and practical environment for comparing the results acquired when new metrics are proposed and applied to symbolic databases, and then, detect what functions get better results for this kind of problem.

2.1 Symbolic Objects

A number of different definitions of symbolic objects are available in the literature. This paper will follow those given by [3], [5] and [10]: symbolic objects are defined by a logical conjunction of events linking values and variables in which the variables can take one or more values and all objects need not be defined by the same variables.

An event is a value–variable pair that links feature variables and feature values of objects. For example, e = [color = {white; red}] is an event indicating that the color variable takes either a white or a red value. A symbolic object is a conjunction of events belonging to a particular object. For example, s = [color = {green; red} $\wedge$ height = [160,190]] is a symbolic object having the following properties: (a) color is either green or red and (b) height ranges between 160 and 190. Based on [3], the symbolic objects can be represented by a vector of features x=({green; red},[160; 190]).

3 Proposed Algorithm

The algorithm implemented here is according to the specifications of FSCM proposed by [8]. See Table 1 for details.

Step 1 consists of initializing the cluster centers randomly. Each group has N/K individuals, where N is the total number of individuals in the database. On step 2.1, the membership matrix W is made of N rows and K columns, each element w_{ij} indicates the degree of membership for the individual with index i in the cluster with index j. The operator $\|.\|$ gives the distance value of an individual X_i to the cluster center Z_j. On step 2.2, Z_{jp} is the center of the cluster j related to the feature p, and contains the e_{kp} frequency of the related event A_{kp} for all symbolic features p in the database. The operator $\theta \in \{0,1\}$ and $\theta=1$ if the feature p of the individual with index i is equal to the event k, otherwise $\theta = 0$. All distances values are normalized in order to avoid overflow. More precise details about how these steps are exactly executed are available in [8].

Table 1. Implemented Algorithm

Input parameters: k (number of groups), m (scalar > 1), Ni (number of iterations), Nr (number of executions)	
Step	**Procedure**
1	Choose cluster centers randomly
2	Repeat until converging or Ni is reached
2.1	Compute the membership matrix W, according to the formula: $w_{ij} = \begin{cases} 1 / \sum_{q=1}^{k} \left(\frac{\lVert X_i - Z_j \rVert}{\lVert X_i - Z_q \rVert} \right)^{2/(m-1)} & , q \neq j \;\&\; q = 1,2,..,k \\ 1, & \end{cases}$ *where* $\lVert X_i - Z_j \rVert = \sum_{p=1}^{\text{n° of features}} \sum_{k=1}^{\text{n° of events}} D(X_{ip}, A_k) * e_{kp} \mid j$
2.2	Compute new cluster centers, according to the formula: $Z_{jp} = \lfloor (A_{1p}, e_{1p}), (A_{2p}, e_{2p}), \ldots, (A_{kp}, e_{kp}) \rfloor \quad p = 1,2,\ldots, n° \text{ of features}$ *where* $e_{kp} \mid j = \sum_{i=1}^{\text{n° of individuals}} w_{ij}^{m} * \theta$
3	Store the membership matrix W of this execution
4	Go to step 1 and repeat the algorithm until Nr is reached
5	Give as output the membership matrix W that minimizes the adequacy criterion

As the initialization of the cluster centers is random, the parameter *Nr* is used to indicate the number of times the algorithm must be executed. On step 3, the membership matrix *W* computed by the actual execution is stored. At the end of the algorithm (step 5) all *W* matrixes are compared and the output that minimizes the adequacy criterion is returned. The adequacy criterion is defined by the following formula:

$$J(Z,W,X) = \sum_{j=1}^{k} \sum_{i=1}^{n} w_{ij}^{m} D^2(X_i, Z_j). \tag{1}$$

The distance function *D* forms the core of the algorithm. It codifies the dissimilarity function to be analyzed. For comparison purposes, different metrics were analyzed: (function 1) Gowda e Diday [10], (function 2) Mali e Mitra [19], (function 3) Ichino e Yaguchi [16] e (function 4) De Carvalho [4]. Besides these, an improved version (function 5) of the Ichino e Yaguchi's function, proposed by De Carvalho[4], was also analyzed.

The algorithm was implemented on Visual C++ version 6.0. Five symbolic databases settings on .sds format were used as input. This file format is adopted by SODAS (Symbolic Official Data Analysis System, http://www.ceremade.dauphine.fr/~touati/

sodas-pagegarde.htm) that provides an appropriate environment to treat symbolic data and implements several methods (decision tree, descriptive statistics, principal component analyses, etc.) for SDA.

4 Experiments

To evaluate the dissimilarity metrics mentioned before, the proposed algorithm was executed under five symbolic representatives databases: microcomputers (Table 2), fat and oil (Table 3), fishes (Table 4) and waves (Table 5). The first two are well know in the literature; Table 2 consists of 12 individuals, two monovalued qualitative variables and three quantitative variables; Table 3 consists of 8 individuals, one multivalued qualitative variable and four quantitative variables. The third database is a result of a study performed on French Guyana about fish contamination [2]. It consists of 12 individuals and 13 interval variables. The fourth database is an example provided by SODAS and consists of 30 individuals and 21 variables of interval type. The fifth database is the zoo data [1] and consists of 100 instances of animals with one character attribute corresponding to the number of legs and 15 boolean features corresponding to the presence of hair, feathers, eggs, milk, backbone, fins, tail; and whether airborne, aquatic, predator, toothed, breathes, venomous, domestic, catsize.

Table 2. Microcomputer Database

Microcomputer	Display	RAM	ROM	MP	Keys
0 Apple II	COLOR TV	48 K	10 K	6502	52
1 Atari 800	COLOR TV	48 K	10 K	6502	57-63
2 Commodore VIC 20	COLOR TV	32 K	11-16K	6502A	64-73
3 Exidi sorcerer	B&W TV	48 K	4 K	Z80	57-63
4 Zenith H8	BUILT-IN	64 K	1 K	8080A	64-73
5 Zenith H89	BUILT-IN	64 K	8 K	Z80	64-73
6 HP-85	BUILT-IN	32 K	80 K	HP	92
7 Horizon	TERMINAL	64 K	8 K	Z80	57-63
8 Sc. Challenger	B&W TV	32 K	10 K	6502	53-56
9 Ohio Sc. II Series	B&W TV	48 K	10 K	6502C	53-56
10 TRS-80 I	B&W TV	48 K	12 K	Z80	53-56
11 TRS-80 III	BUILT-IN	48 K	14 K	Z80	64-73

Table 3. Fat and Oil Database

Name	Gravity	Freezing point	Io. value	sa. value	m. f. acids
0 Linseed oil	0.930-0.935	-27 to -8	170-204	118-196	L,Ln,O,P,M
1 Perilla oil	0.930-0.937	-5 to -4	192-208	188-197	L,Ln,O,P,S
2 Cotton-seed	0.916-0.918	-6 to -1	99-113	189-198	L,O,P,M,S
3 Seaame oil	0.920-0.926	-6 to -4	104-116	187-193	L,O,P,S,A
4 Camellia	0.916-0.917	-21 to -15	80-82	189-193	L,O
5 Olive-oil	0.914-0.919	0 to 6	79-90	187-196	L,O,P,S
6 Beef-tallow	0.860-0.870	30 to 38	40-48	190-199	O,P,M,S,C
7 Lard	0.858-0.864	22 to 32	53-77	190-202	L,O,P,M,S,Lu

It is important to emphasize that all quantitative data is treated as interval data, even those of unique value are treat as a special type of interval where the upper bound is equal to the lower bound. For the fishes, waves and zoo databases there is also a class label represented by a qualitative variable; however, this variable was not used in the algorithm, it was used only to specify the value of the parameter *K*.

For the microcomputers and fat-oil databases the values 5 and 3 were chosen, respectively, as values for the parameter *K* (according to the literature). On the fishes, waves and zoo databases these values were 4, 3 and 7, respectively, known a priori from the class variable.

Table 4. Fishes Dabatase

Name	Long	Poid	Musc	Inte	...	Estomac/ Musc	Feeding
0 Ageneiosusbrevi	22.50-35.50	170.00-625.00	1425.00-5043.00	333.00-2980.06	...	0.00-0.55	Carni-vores
1 Cynodongibbus 1	19.00-32.00	77.00-359.00	2393.00-8737.00	0.00-2653.00	...	0.20-1.24	Carni-vores
2 Hopilasaimara 1	25.50-63.00	340.00-5500.00	1269.65-5937.00	454.50-2181.00	...	0.09-0.40	Carni-vores
3 Potamotrygonhys	20.50-45.00	400.00-6250.00	644.00-936.46	0.00-921.44	...	0.00-0.50	Carni-vores
4 Leporinusfascia	18.80-25.00	125.00-273.00	1231.00-2013.00	0.00-0.00	...	0.12-0.17	Omni-vores
5 Leporinusfreder	2.00-24.50	290.00-350.00	255.00-1085.00	45.00-261.20	...	0.13-0.58	Omni-vores
6 Dorasmicropoeus	19.20-31.00	128.00-505.00	813.00-1947.00	0.00-1430.82	...	0.00-0.79	Detri-tivores
7 Platydorascostas	13.70-25.00	60.00-413.00	317.00-923.00	206.62-649.67	...	0.00-0.61	Detri-tivores
8 Pseudoancistrus	13.00-20.50	55.00-210.00	93.00-114.00	0.00-216.95	...	0.49-1.36	Detri-tivores
9 Semaprochilodus	22.00-28.00	330.00-700.00	293.52-556.61	143.19-933.00	...	0.00-1.25	Detri-tivores
10 Acnodonoligacan	10.00-16.20	34.90-154.70	26.00-115.00	0.00-95.00	...	0.23-5.97	Herbi-vores
11 Myleusrubripini	12.30-18.00	80.00-275.00	8.00-38.00	0.00-0.00	...	0.31-4.33	Herbi-vores

Table 5. Waves Database

Name	Position 1	Position 2	Position 3	Position 4	Position 5	...	Position 21	Form
0 wave_1_3	-2.99:2.63	-1.77:2.56	-2.11:2.69	-1.79:3.29	-1.05:3.29	...	-2.89:2.50	wave_1
1 wave_1_1	-2.09:2.83	-2.94:2.14	-2.53:2.28	-1.97:2.38	-2.34:2.81	...	-2.76:2.72	wave_1
2 wave_1_6	-1.70:2.89	-2.14:2.90	-0.68:3.30	-0.61:3.77	-0.57:4.06	...	-2.45:2.96	wave_1
3 wave_1_9	-2.36:2.69	-1.61:3.31	-0.69:4.67	-0.17:5.45	0.09:5.56	...	-3.03:2.14	wave_1
4 wave_1_5	-2.27:2.33	-2.41:2.97	-1.27:3.16	-0.77:3.77	-0.72:4.50	...	-2.85:2.91	wave_1
5 wave_1_7	-3.79:1.99	-1.76:3.60	-0.64:3.56	-0.30:4.37	-0.57:5.53	...	-2.34:2.48	wave_1
...	...	...	...	...	...	...	...	...

Table 6. Zoo Database

Name	Hair	Feathers	Eggs	Milk	Airborne	...	Catsize	Type
0 aardvark	1	0	0	1	0	...	1	1
1 antelope	1	0	0	1	0	...	1	1
2 bass	0	0	1	0	0	...	0	4
3 bear	1	0	0	1	0	...	1	1
4 boar	1	0	0	1	0	...	1	1
5 buffalo	1	0	0	1	0	...	1	1
...	...	...	...	...	...	...	...	...

5 Results

For analysis purposes, the best group configuration generated by each function was chosen. In addition, the adequacy criterion AC (see Tables 7-11) was also chosen. The AC criterion is computed according to the formula (1). By analyzing this data it can be observed that the functions that reached the lowest values for AC were (function 5), (function 3), (function 4), (function 1) and (function 2), in crescent order. It is important to stress that this result is kept the same for all databases evaluated. The results show that the function that had the best result, i.e., the lower value for the adequacy criterion, hence, greater connectivity between members of same group was the one proposed by Ichino and Yaguchi adapted by De Carvalho[4].

However, none of the results obtained reaches the perfect grouping for the studied case (see Table 12), i.e., the functions did not allocate correctly in the same group all individuals that belong to the same class. For the fishes (Table 4), waves (Table 5) and also zoo (Table 6) databases, this information was supplied a priori by the class label. For the microcomputers (Table 2) and fat-oil (Table 3) databases, the clustering information was supplied by the literature.

Table 7. Clustering Results for MicroComputer Database

Group	Funçtion 1	Funçtion 2	Funçtion 3	Funçtion 4	Funçtion 5
1	0,1,7,8	0,1,8,9	0,1,2,7,8	0,1,2,3,6,7,9	0,1,2,3,4,6,7,8
2	4,5,11	4,5,7	4,5	^{1},5	5
3	6	2,6	6	8	9
4	3,9,10	3,10	3,9,10	10	10
5	2	11	11	11	11
AC =	1.96293	5.39487	0.250613	1.02259	0.236831

Table 8. Clustering Results for Fat and Oil Database

Group	Funçtion 1	Funçtion 2	Funçtion 3	Funçtion 4	Funçtion 5
1	0,1,2,3,4,5	0,1,2,3,4,5	2,3,4,5	0,1,2,3,4,5	0,1,2,3,4,5
2	7	7	0,1	7	7
3	6	6	6,7	6	6
AC =	4.36096	7.26356	0.225571	0.72593	0.194956

Table 9. Clustering Results for Fishes Database

Group	Function 1	Function 2	Function 3	Function 4	Function 5
1	0,1,2,3,4,5,6,8,9	0,1,2,3,4,5,6,7,9	0,2,3,4,5,6,7,8,9	0,1,2,3,4,5,6,7,9	0,1,2,3,4,5,6,7,9
2	10	10	10	10	10
3	11	11	11	11	11
4	7	8	1	8	8
AC =	18.7793	20.1549	0.316695	0.967519	0.117048

Table 10. Clustering Results for Waves Database

Group	Function 1	Function 2	Function 3	Function 4	Function 5
1	0,1,2,3,4,5,6,7,8,9,11,12,16,17,18,20,21,23,24,25,26,28,29	0,1,6,9,20,21,23,24,25,26,27,28,29	0,1,6,9,20,21,23,28,29	0,1,6,9,20,21,23,25,26,28,29	3,5,7,8,12,13,14,15,16,17,18,19
2	14,19	7,8,10,11,12,14,15,16,17,18,19	2,3,4,5,7,8,12,13,14,16,17,18,19	2,3,4,5,7,8,12,13,14,15,16,17,18,19	0,1,6,9,20,21,23,25,26,28,29
3	10,13,15,22,27	2,3,4,5,13,22	10,11,15,22,24,25,26,27	10,11,22,24,27	2,4,10,11,22,24,27
AC =	13.8116	19.8523	0.106495	0.28663	0.0333606

Table 11. Clustering Results for Zoo Database

Group	Function 1	Function 2	Function 3	Function 4	Function 5
1	24,29,38,39,41,50,80,87,96,98	24,29,38,39,41,50,80,87,96,98	24,29,38,39,41,50,80,87,96,98	24,29,38,39,50,80,87,96,98	24,29,38,39,41,50,80,87,96,98
2	90	89	19,65,73,74	79,90	19,65,73,74
3	25,51,79,88,90	1,5,6,9,17,22,26,27,28,30,31,34,35,54,64,69,83,92,93,95	1,5,6,9,17,22,26,27,30,31,34,35,54,64,69,83,89,92,93,95	1,5,6,9,17,22,26,27,30,31,34,35,54,64,69,83,92,93,95	25,51,79,88,90
4	11,16,20,21,23,32,36,40,42,55,56,57,58,70,77,78,82,86,94,99	11,16,20,21,23,32,36,40,42,55,56,57,58,70,77,78,82,86,94,99	11,16,20,21,23,32,36,40,42,55,56,57,58,70,77,78,82,86,89,94,99	11,16,20,21,23,32,36,40,42,55,56,57,58,70,77,78,82,86,89,94,99	11,16,20,21,23,32,36,40,42,55,56,57,58,70,77,78,82,86,89,94,99
5	2,7,8,12,18,33,37,51,59,60,61,72,75,79,81,85,91	2,7,8,12,18,33,37,51,59,60,61,72,75,79,81,85,90,91	2,7,8,12,18,33,37,59,60,61,72,75,81,85,91	2,7,8,12,18,33,37,51,59,60,61,72,75,81,85,91	2,7,8,12,18,33,37,59,60,61,72,75,81,85,91
6	13,14,15,25,45,52,71,76,84,88	13,14,15,25,45,52,71,76,84,88	13,14,15,45,52,71,76,84	13,14,15,25,41,45,52,71,76,84,88	13,14,15,45,52,71,76,84
7	0,3,4,10,19,28,43,44,46,47,48,49,53,62,63,65,66,67,68,73,74,97	0,3,4,10,19,43,44,46,47,48,49,53,62,63,65,66,67,68,73,74,97	0,1,3,4,5,6,9,10,17,22,26,27,28,30,31,34,35,43,44,46,47,48,49,53,54,62,63,64,66,67,68,69,83,92,93,95,97	0,3,4,10,19,28,43,44,46,47,48,49,53,62,63,65,66,67,68,73,74,97	0,1,3,4,5,6,9,10,17,22,26,27,28,30,31,34,35,43,44,46,47,48,49,53,54,62,63,64,66,67,68,69,83,92,93,95,97
AC =	3.97023	6.77761	0.951918	1.83077	0.25173

Table 12. Best Clustering for Analyzed Databases

Cluster	Microcomputers database	Fat and oil database	Fishes database	Waves database	Zoo Database
1	0,1,3,9,10	0	0,1,2,3	0,1,2,3,4,5, 6,7,8,9	0, 1, 3, 4, 5, 6, 9, 10, 17, 19, 22,26, 27, 28, 30, 31, 34, 35, 43, 44, 46, 47, 48, 49,53, 54, 62, 63, 64,65, 66, 67, 68, 69,73, 74, 83, 92, 93, 95,97
2	4,5,11	1,2,3,4,5	4,5	10,11,12,13,14, 15,16,17,18,19	11, 16, 20, 21, 23, 32, 36,40, 42, 55, 56, 57, 58,70, 77, 78, 82, 86, 94, 99
3	2,8	6,7	6,7,8,9	20,21,22,23,24, 25,26,27,28,29	61, 75, 79, 89, 90
4	6	-	10,11	-	2, 7, 8, 12, 18, 33,37, 59, 60, 72, 81, 85, 91
5	7	-	-	-	25, 51, 88
6	-	-	-	-	24, 29, 38, 39, 41, 50, 87, 96
7	-	-	-	-	13, 14, 15, 45, 52,71, 76, 80, 84, 88

6 Conclusions and Future Works

More research is still needed in order to obtain a function that better approximates the perfect clustering.

The implemented algorithm adopted a pattern for the input data, which, together with its modularity, allows it to be used in future comparisons of new dissimilarity metrics on any symbolic databases, all that without great efforts or changes. This way, it is easier and faster to analyze clusters generated by applying different metrics.

Future work may include: definition of a new structure for class prototypes, and utilization of graphics tools to show how individuals are arranged into the clusters on each iteration.

References

1. Blake, C.L., Merz, C.J.: UCI repository of machine learning databases, University of California, Irvine, Dept. of Information and Computer Sciences, 1998. Available from <http://www.ics.uci.edu/~mlearn/MLRepository.html>.
2. Bobou, A., Ribeyre, F.: Mercury in the food web: Accumulation and transfer mechanisms. Metal Ions in Biological Systems. Marcel Dekker, New York, 1998, pp. 289–319.
3. Bock , H.H., Diday, E.: Analysis of Symbolic Data, Exploratory Methods for Extracting Statistical Information from Complex Data. Springer, Heidelberg, 2000.
4. De Carvalho, F. A. T.: Proximity coefficients between boolean symbolic objects, In: Diday, E. et al. (Eds.), New Approaches in Classification and Data Analysis, Springer, Heidelberg, 1994, pp. 387-394.

5. Diday, E.: The symbolic approach in clustering and related methods of data analysis, Classification Methods of Data Analysis, North-Holland, Amsterdam, 1988, pp. 673–684.
6. Diday, E., Brito, M. P.: Symbolic cluster analysis, Conceptual and Numerical Analysis of Data, Springer, Heidelberg, 1989, pp. 45–84.
7. Diday, E., Simon. J. J.: Clustering analysis. In: Fu, K.S. (Ed.), Digital Pattern Recognition. Springer, Heidelberg, 1976,pp. 47–94.
8. El-Sonbaty, Y., Ismail, M. A.: Fuzzy clustering for symbolic data. IEEE Transactions on Fuzzy Systems, v. 6, 1998, pp. 195–204.
9. Gordon, A.D.: An iterative relocation algorithm for classifying symbolic data. In: Gaul, W. et al. (Eds.), Data Analysis: Scienti.c Modeling and Practical Application. Springer, Heidelberg, 2000, pp. 17–23.
10. Gowda, K. C., Diday E.: Symbolic clustering using a new dissimilarity measure, Pattern Recognition Letters,1991, v.24, n.6, p.567-578.
11. Gowda, K. C., Diday, E.: Symbolic clustering using a new similarity measure, IEEE Transactions on Systems, Man and Cybernetics, v. 22, 1992, pp. 368–378.
12. Gowda, K. C., Ravi, T. R.: Divisive clustering of symbolic objects using the concepts of both similarity and dissimilarity, Pattern Recognition Letters, v. 28, n. 8, 1995a, pp. 1277–1282.
13. Gowda, K. C., Ravi, T. R.: Agglomerative clustering of symbolic objects using the concepts of both similarity and dissimilarity, Pattern Recognition Letters, v.16, 1995b, pp. 647–652.
14. Gowda, K. C., Ravi, T. R.: Clustering of symbolic objects using gravitational approach, IEEE Transactions on Systems, Man and Cybernetics, n. 29, v. 6, 1999a, pp. 888–894.
15. Gowda, K. C., Ravi, T. R.: An ISODATA clustering procedure for symbolic objects using a distributed genetic Algorithm, Pattern Recognition Letters, v. 20, 1999b, pp. 659–666.
16. Ichino, M., Yaguchi, H.: Generalized minkowski metrics for mixed feature-type data analysis, IEEE Transactions on Systems, Man, and Cybernetics, v. 24, n. 4, 1994, p. 698-708.
17. Jain, A. K., Dubes, R. C.: Algorithms for Clustering Data. Prentice-Hall, Englewood Cli.s, NJ, 1988.
18. Jain, A. K., Murty, M. N., Flynn, P. J.: Data clustering: A review, ACM Computing Surveys, v. 31, n. 3, 1999, pp. 264–323.
19. Mali, K., Mitra, S.: Clustering and its validation in a symbolic framework, Pattern Recognition Letters, 2003, v. 24, n. 14, pp. 2367-2376.

Estimating User Location in a WLAN Using Backpropagation Neural Networks

Edgar A. Martínez[1], Raúl Cruz[1], and Jesús Favela[2]

[1] Universidad Tecnológica de la Mixteca, Oaxaca, México
[2] Departamento de Ciencias de la Computación, CICESE, Ensenada, México
ealbertomtz@prodigy.net. mx,
rcruz@mixteco.utm.mx, favela@cicese.mx

Abstract. Context-aware computing refers to an application's ability to adapt to changing circumstances and respond based on the context of use. The estimation of user location is crucial to many context-aware applications. In this paper we propose a technique to infer user location in a wireless LAN inside buildings based on backpropagation neural networks. The strengths of the radio-frequency (RF) signals arriving from several access points in a wireless LAN are related to the position of the mobile device. Estimating the position of the mobile device from the RF signals is a complex inverse problem since the signals are affected by the heterogeneous nature of the environment. Backpropagation neural networks represent a viable alternative to tackle this problem given their property of generalizing from examples. Experimental results provide an average distance error of 1.87 meters from the real location, which is considered to be adequate for many applications. A simple context-aware application is presented as an example. The estimation errors obtained are similar to those using k-nearest neighbors; however the approach presented here uses less memory, an important concern for handheld devices with limited storage capacity an operating on relatively slow WLANs.

1 Introduction

Context-aware computing refers to an application's ability to adapt to changing circumstances and respond based on the context of use. A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task [1]. Among the main types of contextual information considered relevant are identity, time, activity, and location, which can be used to derive additional, related information.

Context is difficult to use for several reasons [1]. First, capturing primary contextual information requires the use of sensors and computing devices. Context must be abstracted to make sense to the application, for instance, the ID of a mobile user must be abstracted into the user's name or role. Finally, context is dynamic, i.e., a mobile tour guide must update its display as the user moves, which requires the tracking the user's location by gathering information from multiple sensors, and using techniques that estimate the user's location or guess the route that a user will follow, which may introduce uncertainty.

A number of location-based applications have been implemented in recent years, such as the Guide system that provides city visitors with a hand-held context-aware tourist guide. The ordering of the tour recommended by this system changes dynamically when the visitor stays at a location longer than anticipated [2]. Safe & Sound is a location-tracking system that allows parents to monitor their children's position [3].

Undoubtedly, location is important to understand the context of mobile users. Location is useful to infer other contextual variables that a system might use to provide services and information to mobile users.

Several systems and methods for location estimation have been proposed. Outdoors systems like GPS [4] can locate mobile computers with accuracy but they are ineffective indoors. On Indoor location techniques, proprietary infrastructure based systems such as infrared [5] and ultrasound [6] provide location information with a high infrastructure cost. Also, location video based systems like in [7] utilize pattern matching techniques but they need large amounts of training to achieve accurate results. On RF-based location, we find systems that use an existing wireless LAN infrastructure and signal strength to estimate position like RADAR [8], Nibble [9] and techniques developed by Smailagic *et al.* [10].

The method we propose in this work is based on backpropagation neural network models. This approach uses an existing wireless LAN infrastructure to measure the signal strengths from the access points (APs) to the mobile device. A backpropagation neural network is trained to map the signal strength to 2D coordinates.

The paper is organized as follows: In section 2, we survey related work in the field of location-estimation. In section 3, we describe the architecture of the backpropagation neural network and its use to estimate user location. Section 4 presents our experimental testbed. The results obtained are presented in section 5. Finally, in Section 6 we illustrate the use of the location estimation technique by presenting a location-aware system to help people to be oriented inside a building and upon which, more sophisticated location-aware applications can be developed.

2 Estimating User Location

Estimating the location of a user has been a subject of considerable attention in context-aware computing in recent years. We can classify location and tracking approaches in the following categories [8]: IR-based, indoor RF-based, wide-area cellular-based, ultrasound-based, floor-sensing, and vision-based systems.

In IR-based systems, the first location-based computer applications reported in the literature was the Active Badge. This system used infrared signals emitted every 10 seconds by badges and received by infrared sensors located in a building that picked up the unique identifiers and relay these to a location management system. These systems can only locate people wearing a small infrared badge [5]. Besides, IR-based systems have significant installation and maintenance costs and the range of the infrared signal is limited to a few meters. Similar IR-based tracking systems were developed for Xerox's ParcTab [11].

In wide-area cellular-based systems, advances in Global Positioning Systems (GPS) allow mobile computers to determine its location with considerable accuracy. GPS use triangulation of the signals received from multiple satellites to determine location with an approximate accuracy of 10 meters [4]. Technological alternatives to determinate location in wide areas involve measuring the signal's attenuation, the angle of arrival (AOA), and/or the time difference of arrival (TDOA). The disadvantage of wide-area cellular-based systems is that they are ineffective indoors because buildings block or reflect RF signals.

Other techniques involve the use of ultrasound. An example is the Active Bat [6], which uses an ultrasound time-of-flight lateration technique to provide the physical positioning. Other non-cellular approaches include the Smart Floor [12] where embedded pressure sensors capture footfalls and the system uses the data for tracking and recognizing people. Computer vision technology with the use of multiple cameras has also been proposed to track user location [7]. Such systems, however, have line of sight problem and work with only a small number of persons in a room.

Of particular interest are location estimation techniques that make use of an existing wireless LAN infrastructure, since they have better scalability and less installation and maintenance costs than ad-hoc solutions. Also, these systems are effective for indoor environments. These methods use the RF signal strength, to estimate the distance between a mobile device and several access points of the wireless LAN. A signal propagation model could be used to estimate this distance, but rather complex models would be required, since the signal is affected by the presence of walls, furniture and other people and devices. To work around this complexity, empirical methods have been advanced. In these methods the strength of the RF signal is measured at predefined locations and used to train a pattern recognition model that can then be used to estimate the user's location.

The RADAR location system uses an IEEE 802.11b wireless LAN and an empirical method based on the nearest neighbor in signal space algorithm [8]. Similarly, the Nibble system uses the signal-to-noise ratio, which is more stable than signal strength to compute the distance to the access point. Nibble uses a Bayesian network to estimate the probability of the mobile being at one of a set of discrete locations [9]. Alternative methods, developed in Carnegie Mellon University [10], are CMU-TMI which use triangulation, mapping and interpolation; and CMU-PM which uses pattern matching similar to [8].

3 Using Neural Networks to Estimate User Location

In this work, we use backpropagation neural networks trained to map the signal strength to 2D coordinates. A backpropagation neural network, or multi-layer perceptron, is a supervised non-parametric model that learns from a training set by adjusting the weights that shape the strength of the signals propagated between processing units, or neurons. Once trained, the neural network can be used to classify incoming patterns into labeled classes.

The architecture of a backpropagation neural network is made of two or more layers of processing nodes with each of the nodes of the i-layer connected to each of the nodes in the i+1-layer. The connections have different strengths (or weights) that represent the influence that a particular node has in a node of a subsequent layer. Each node computes an activation value that is the result of applying a non-linear function (typically the sigmoid function, when the errors are assumed to be gaussian) to the sum of the products of the activation weights of the previous layer, times the weight that connect each of those nodes with the unit performing the computation.

A network that is presented with an input pattern in the first layer will then generate a pattern in the output (or last) activation layer. The most important aspect of neural networks though, is not how they compute these output patterns, but rather how they learn from a set of examples. The backpropagation learning algorithm calculates how the nodes in the internal layers will be penalized (adjusted) when presented with a training pair (a tuple of input and output patterns).

Fig. 1 shows a simplified version of the architecture of the neural network we have used. The signal strength from each access point is presented to the input layer. We have used a single hidden layer and a two-node output that represents the X and Y coordinates of the user's location.

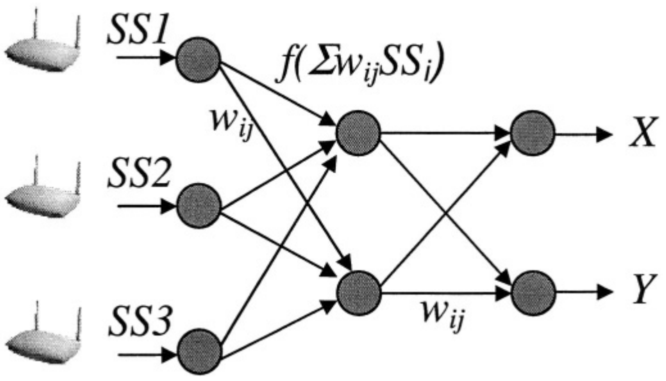


Fig. 1. Architecture of the neural network used to estimate the location of mobile users

We used the logistic function as the activation function for the hidden layer and the identity function for the output layer. Input and output data, for training and testing the neural network, were scaled so they fall in the range [-1, 1].

4 Experimental Testbeds

Our testbed was deployed on the second floor of the Computer and Electronics Institute of the Universidad Tecnologica de la Mixteca in Oaxaca, Mexico. The dimensions of the floor are 40x20 meters, which include 28 rooms. A wireless Local

Area Network (wLAN) based on the IEEE 802.11b standard is running in it. Five access points cover the entire floor. Two access points are Apple's Airport and three are Orinoco's AP-200. The wireless station is a Pentium-based laptop computer running Windows XP. The laptop is equipped with an Orinoco Silver PC wLAN card.

The network operates in the 2.4 GHz license-free ISM (Industrial, Scientific and Medical) band and has a range of 160 m, 50 m, and 25 m, respectively, for open, semi-open, and closed office environments. The 2.4 GHz ISM band is divided into 13 channels and five of those are used: channels 1, 3, 7, 9 and 11 in order to minimize the interference.

To train the neural network we measured the signal strength from the five access points. We did this on 154 different locations within the building. At each location and for each direction: north, south, east, and west, a laptop computer with the network interface card was set to take measurements for 15 seconds with a frequency of one sample every 2 seconds. At all points we recorded the signal strength from at least four access points. The signal was considered to be in its lowest level (-102 dBm) when the signal from one of the access points was too weak to be measured.

To read the measurements we used Orinoco's Client Manager v2.9 for Windows XP. A total of 616 measurements points were collected during a whole day.

5 Results

The basic backpropagation algorithm adjusts the weights in the steepest descent direction, that means that the performance function decreases rapidly but it does not necessarily converges. Therefore we decided to use conjugate gradient algorithms, which perform a search along conjugate directions, converging generally faster [13]. In this work we used four of these algorithms: the Polak-Ribiere updates (traincgp) [14], Scaled Conjugate Gradient (trainscg) [15], Fletcher-Reeves updates (traincgf) [14], and Powell-Beale restarts (traincgb) [16].

Besides the conjugate gradient algorithms we also used Quasi-Newton Algorithms, which are based on Newton's method, but which do not require the calculation of second derivatives [17]. In this class of algorithms we used the one step secant backpropagation (trainoss) [18], the Broyden, Fletcher, Goldfarb and Shanno updates (trainbfg) [17], and the Levenberg-Marquardt backpropagation algorithm (trainlm) [13].

We carried out experiments with four network architectures for each of the training algorithm mentioned above, with 4, 6, 8 and 16 hidden units. We trained the network with 432 patterns and used remaining 184 as test set. To compare the results we calculated the average distance error:

$$\textbf{Distance Error} = \frac{1}{P}\sum_{p=1}^{P}\sqrt{(tx_p - ox_p(w))^2 + (ty_p - oy_p(w))^2}\,, \qquad (1)$$

where $\mathbf{ox_p(w)}$ and $\mathbf{oy_p(w)}$ are the x and y coordinates obtained by the network and $\mathbf{tx_p}$ and $\mathbf{ty_p}$ are the correct target values. P is the number of tests or training patterns.

We stopped the network's training when the gradient norm was less or equal to 2E-4 or when the average distance error over the test set had a tendency to increase. We used the recorded signal strengths as training and test sets. Table 1 shows the results for the test set.

Table 1. Results over the test set (in meters)

Training Algorithm	Average error with 4 hidden units	Average error with 6 hidden units	Average error with 8 hidden units	Average error with 16 hidden units
Trainlm	2.4912	2.3639	2.1726	2.4303
Trainscg	2.4517	2.2592	2.2492	2.22
Trainoss	2.8271	2.2947	2.2716	2.2573
Traincgp	2.8194	2.2827	2.3145	2.2733
Traincgf	2.4242	2.2489	2.257	2.2242
Traincgb	2.4089	2.2837	2.2979	2.2666
Trainbfg	2.5127	2.272	2.2393	2.2704

From Table 1 we see that the best architectures for trainlm, trainscg, trainoss, traincgp, traincgf, traincgb and trainbfg have 8, 16, 16, 16, 16, 16 and 8 hidden units, respectively. We can also observe that the best overall algorithm was trainlm, with 8 hidden units. It produced an average error of 2.1726 meters from the real location.

5.1 Signal to Noise Ratio as Input Variable

We experimented with signal to noise ratio (SNR) as an alternative input to the neural network. We used SNR signals to improve the network performance, since these signals tend to be more stable than the signal's strength. We trained the networks using again the algorithms from Table 1. Table 2 shows the results for the best architectures selected for each algorithm.

Table 2. Results over the test set using SNR as network input

Training Algorithm	Network Architecture	Average error (meters)	CDF in 2 meters (%)	CDF in 3 meters (%)	CDF in 4 meters (%)
Trainlm	5 – 8 - 2	2.1953	0.53	0.80	0.90
Trainscg	5 – 16 - 2	2.1727	0.54	0.76	0.89
Trainoss	5 – 16 -2	2.1518	0.53	0.77	0.90
Traincgp	5 – 16 - 2	2.1533	0.55	0.75	0.89
Traincgf	5 – 16 -2	2.0947	0.54	0.79	0.91
Traincgb	5 - 16 - 2	2.19	0.51	0.78	0.91
Trainbfg	5 – 8 - 2	2.314	0.53	0.75	0.90

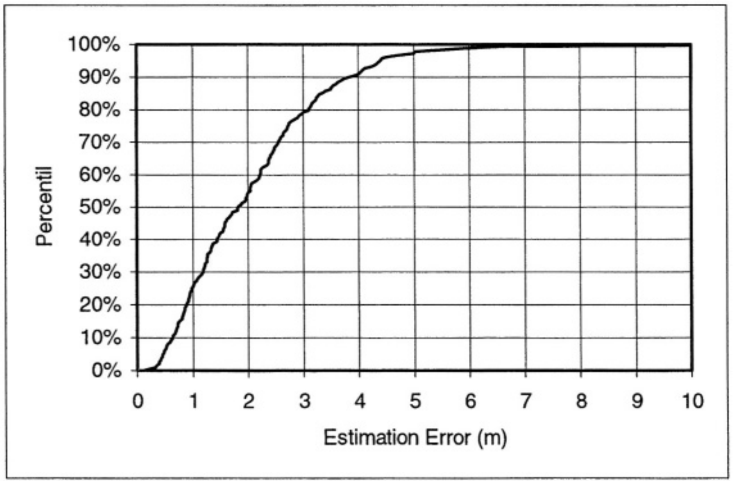


Fig. 2. Cumulative Distribution Function (CDF) of the traincgf algorithm with 16 hidden units

We can observe from Table 2 that the traincgf algorithm improves the network performance achieving an average error of 2.0947 meters from the real location. Figure 2 shows the cumulative distribution function (CDF) for this algorithm. This graph shows the percentage of patterns that fall within a given distance. It can be seen, for instance, that 79% of the patterns are within 3 meters of their target.

5.2 Reducing the Average Distance Error Using a People-Oriented Criterion

In a context-aware system we can take into account the fact that people move smoothly seldom exceeding a speed of 1.5 m/s when walking in closed public spaces. Thus, we proposed a slight modification to our algorithm which consists of taking four samples and estimating the location of the person of each of them, to then discard those estimations that are more 1.5 m away from the the mean of the other three estimations. We calculated user location using this criterion using the same data sets. The results obtained are shown in Table 3. We observed that the traincgf algorithm reaches an average distance error of 1.8684 meters from the real location, a 12% improvement in accuracy.

Table 3. Results of the test set using the people oriented criterion

Training Algorithm	Network Input	Average error (m)	CDF in 2 meters (%)	CDF in 3 meters (%)	CDF in 4 meters (%)
Trainlm	Signal Strength	1.9168	0.60	0.84	0.95
Traincgf	SNR	1.8684	0.58	0.84	0.95

5.3 Comparison with the K-Nearest Algorithm

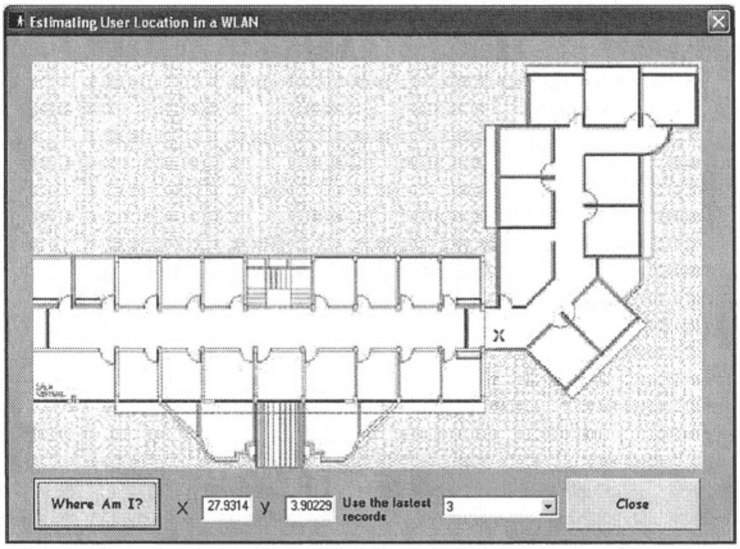
We compared our best results obtained in section 5.2 against the k-nearest algorithm [8][19] applied over the same training and test data sets (SNR) and using the estimation criterion mentioned above. The comparison is shown in Table 4, with the best result obtained with k-NN (8-NN) and the fastest and simplest one (1-NN).

Table 4. Comparison between the neural network and k-nearest location estimation techniques

Technique	Average error (m)	CDF in 2 meters (%)	CDF in 3 meters (%)	CDF in 4 meters (%)
Neural Network	1.8684	0.58	0.84	0.95
8-nearest neighbor	1.9004	0.60	0.86	0.95
Nearest neighbor	1.9504	0.66	0.85	0.92

6 A Sample Application

To illustrate the use of the location-estimation technique proposed in this work we present a simple location-aware system. The application uses the previously trained neural network to estimate the location of a user who moves through the building's floor and displays this information on a map. The interface of this application is shown in Figure 3.



X 27.9314 y 3.90229 Use the lastest records 3

Fig. 3. Location-aware system indicating the estimated location of the user moving through the building

7 Conclusions

In this paper we describe and evaluate a new location estimation model based on backpropagation neural networks to estimate the position of a mobile device in a wireless LAN within a building. The technique only needs a wireless LAN with the IEEE 802.11b standard infrastructure rather than requiring a specialized infrastructure solely for this purpose. The results obtained with the backpropagation neural network are equivalent to the results with the k-nearest algorithm.

Neural networks, however, offer the advantage of being less memory intensive, an important feature given the limitations of PDAs. With the configuration and data we have used, for instance, the nearest neighbor approach will require the storage of 23 times more data than neural networks. This is important not only in terms of storage, but also for the transfer of this data through the wireless network when a user moves to a new floor. In future work we plan to use a Hidden Markov Model to predict location, based not only on the readings from the access points but from the previous location of the user, we believe that this could help us improve the accuracy of our estimation for practical applications such as the one presented. In addition we plan to integrate this technique to context-aware medical applications [20].

References

1. Dey, A.K.: Understanding and Using Context. Personal and Ubiquitous Computing, Vol. 5, No.1. Springer-Verlag, London UK (2001) 4-7
2. Cheverst, K., Davies, N., Mitchell, K., Friday, A.: Experiences of Developing and Deploying a Context-Aware Tourist Guide: The GUIDE Project. In Proc. of Mobile Computing and Networking (2000) 20-31
3. Marmasse, N., Shmandt, C.: Safe & Sound – a wireless leash. In Proc. of CHI 2003, extended abstracts (2003) 726-727
4. Enge, P., Misra, P.: Special Issue on GPS: The Global positioning System. In Proceedings of the IEEE, January (1999) 3-172
5. Want, R., Hoopper, A., Falcao, V., Gibbons, J.: The active badge location system. ACM Trans. on Information Systems, Vol. 10 No. 1 (1992) 91-102
6. Hazas, M., Ward., A.: A novel broadband ultrasonic location system. Proc. of Ubiquitous Computing Conference, Springer LNCS 2498, Goteborg Sweden (2002) 264-280
7. Darrell, T. et al.: Integrated person tracking using stereo, color, and pattern detection. In Proc. of Computer Vision Conf. and Pattern Recognition, IEEE CS Press (1998) 601-608
8. Bahl, P., Padmanabhan, V.N.: RADAR: An in-building RF-based location and tracking system. In IEEE INFOCOM 2000, Tel-Aviv Israel (2000)
9. Castro, P., Chiu, P., Kremenek, T., Muntz, R.: A Probabilistic Room Location Service for Wireless Networked Environments. In Proc. of the 3rd. Intl. Conf. on Ubiquitous Computing: UbiComp 2001, LNCS 2201, Springer, Atlanta GA USA (2001) 18-34
10. Smailagic, A., Siewiorek, D., Anhalt, J., Kogan, D., Wang, Y.: Location Sensing and Privacy in a Context Aware Computing Environment. Pervasive Computing (2001)
11. Want, R., Schillit, B., Adams, N., Gold. R.: The ParcTab Ubiquitous Computing Experiment. Mobile Computing, chapter 2, Luwer Academic Publishers (1996)

12. Orr, R.J., Abowd, G.D.: The Smart Floor: A Mechanism for Natural User Identification and Tracking. In Proc. 2000 Conf. Human Factors in Computing Systems (CHI 2000), ACM Press, New York (2000)
13. Hagan, M.T., Demuth, H.B., Beale, M.H: Neural Network Design, PWS Publishing (1996)
14. Scales, L. E.: Introduction to Non-Linear Optimization. Springer-Verlag, New York (1985)
15. Moller, M. F.: A scaled conjugate gradient algorithm for fast supervised learning. Neural Networks, Vol. 6 (1993) 525-533
16. Powell, M. J. D.: Restart procedures for the conjugate gradient method. Mathematical Programming, Vol. 12 (1977) 241-254
17. Dennis, J. E., Schnabel, R. B.: Numerical Methods for Unconstrained Optimization and Nonlinear Equations. Prentice-Hall, Englewood Cliffs NJ (1983)
18. Battiti, R.: First and second order methods for learning: Between steepest descent and Newton's method. Neural Computation, Vol. 4, No. 2 (1992) 141-166
19. Battiti, R., Villani, A., Le Nath, T.: Neural network model for intelligent networks: deriving the location from signal patterns. In Proc. of the First Annual Symposium on Autonomous Intelligent Networks and Systems, UCLA, May 8-9 (2002)
20. Muñoz, M. A., Rodriguez, M., Favela, J., Gonzalez, V.M., Martinez-Garcia, A.I.: Context-aware mobile communication in hospitals. IEEE Computer, Vol. 36, No. 8 (2003) 38-46

On the Optimal Computation of Finite Field Exponentiation

Nareli Cruz-Cortés, Francisco Rodríguez-Henríquez, and Carlos A. Coello Coello

Computer Science Section, Electrical Engineering Department,
Centro de Investigación y de Estudios Avanzados del IPN,
Av. Instituto Politécnico Nacional No. 2508, México D.F.
nareli@computacion.cs.cinvestav.mx, {francisco, coello}@cs.cinvestav.mx

Abstract. It has been shown that the optimal computation of finite field exponentiation is closely related to the problem of finding a suitable addition chain with the shortest possible length. However, obtaining the shortest addition chain for a given arbitrary exponent is a **NP**-hard problem. Hence in general, we are forced to use some kind of heuristic in order to compute field exponentiation with a semi-optimal number of underlying arithmetic operations. In this paper we present a novel heuristic for that problem which is based on an immune artificial system strategy. The results obtained by our scheme yield the shortest reported lengths for the exponents typically used when computing field multiplicative inverses for error-correcting and elliptic curve cryptographic applications.

1 Introduction

The problem of efficiently compute finite field or group exponentiation is an illustrious mathematical problem with a long and interesting history related to the theoretical optimization problems found in computer science [1]. In addition to its theoretical relevance, field exponentiation has many important practical applications in the areas of error-correcting codes and cryptography. For instance, field or group exponentiation is used in several major public-key cryptosystems such as RSA, Diffie-Hellman and DSA [2].

A finite field is a set of a finite number of elements q where all the basic arithmetic operations such as: addition, subtraction, multiplication, inversion and exponentiation are well defined and can be performed without leaving the field. In several relevant applications the number of field elements can be as high as 2^{512} or more.

Let α be an arbitrary element of a finite field F, and e and arbitrary positive integer. Then, field exponentiation is defined as the problem of finding an element $\beta \in F$ such that the equation $\beta = \alpha^e$ holds. In general one can follow two strategies to obtain an efficient implementation of field exponentiation. One approach is to implement field multiplication, the main building block required for field exponentiation, as efficiently as possible. The other is to reduce the total number of multiplications needed to compute β. In this paper we addressthe latter approach, assuming that arbitrary choices of the base element α are allowed but with the restriction that the exponent e is fixed.

There are a large number of reported algorithms to solve field exponentiation. Reported strategies include: binary, m-ary, adaptive m-ary, power tree and the factor method, to mention just a few [3, 1, 2]. All those algorithms have in common that they strive to keep the number of required field multiplications as low as possible through the usage of a particular heuristic. However, none of those strategies can be considered to yield an optimal solution for every possible scenario and/or application. On the other hand, all the aforementioned heuristics can be mathematically described by using the concept of *addition chains.* Indeed, taking advantage of the fact that the exponents are additive, the problem of computing powers of the base element α, can be directly translated to an addition calculation. This observation leads us to the concept of an *addition chain* for a given exponent e that can be informally defined as follows.

An *addition chain* for e of length l is a sequence U of positive integers, $u_0 = 1, u_1 \ldots, u_l = e$ such that for each $i > 1$, $u_i = u_j + u_k$ for some j and k with $0 \leq j \leq k < i$.

The solely purpose in life of an addition chain is to minimize the number of multiplications required for an exponentiation. Indeed, if U is an addition chain that computes e as mentioned above then for each $\alpha \in F$ we can find $\beta = \alpha^e$ by successively computing: $\alpha, \alpha^{u_1}, \ldots, \alpha^{u_{l-1}}, \alpha^e$.

Let $l(e)$ be the shortest length of any valid addition chain for a given positive integer e. Then the theoretical minimum number of field multiplications required for computing the field operation $\beta = \alpha^e$ is precisely $l(e)$. Unfortunately, the problem of finding an addition chain for e with the shortest length $l(e)$ is an **NP**-hard problem [2]. In order to solve that optimization problem we present in this work an approach based on the Artificial Immune System (AIS) paradigm.

AIS is a relatively new computational intelligence paradigm which borrows ideas from the natural immune system to solve engineering problems. AIS has been successfully applied to solve problems in different areas such as computer and network security, scheduling, machine learning [4, 5] and optimization. Reported optimization problems solved using AIS systems include multimodal, numerical [6], and combinatorial optimization [7]. However to the best of our knowledge, the algorithm presented in this work constitutes the first attempt to use AIS as a heuristic to find optimal addition chains for field exponentiation computations. As a specific design example, we describe how to use our technique for computing field inversion via Fermat's little theorem [8, 9].

The rest of this paper is organized as follows. In Section 2 we describe the strategy followed in this paper in order to find optimal addition chains using an AIS approach. Then, in section 3 we explain how an optimal addition chain can be used in practice to solve finite field multiplicative inversion. A comparison between our results and other reported works is given in Section 4. Finally, in Section 5 some conclusions remarks are drawn.

2 Obtaining Optimal Addition Chains Using an AIS Approach

In this section we describe how the AIS paradigm can be used to solve the problem of finding optimal addition chains. We start this section with a formal definition of an addition chain followed by a brief description of artificial immune systems. We end this section giving a description of the algorithm utilized.

2.1 Definition

An *addition chain* [2] U for a positive integer e of length l is a sequence of positive integers $U = \{u_0, u_1, \cdots, u_l\}$, and an associated sequence of r pairs $V = \{v_1, v_2 \cdots, v_l\}$ with $v_i = (i_1, i_2), 0 \leq i_2 \leq i_1 < i$, such that:

- $u_0 = 1$ and $u_l = e$;
- for each $u_i, 1 \leq i \leq l, u_i = u_{i_1} + u_{i_2}$.

Example 1. Consider the case $e = 415 = (110011111)_2$. Then, the binary addition chain with length $l = 14$ for that e is,

$$U := 1 \rightarrow 2 \rightarrow 4 \rightarrow 8 \rightarrow 16 \rightarrow 32 \rightarrow 64 \rightarrow 128 \rightarrow$$
$$256 \rightarrow 384 \rightarrow 400 \rightarrow 408 \rightarrow 412 \rightarrow 414 \rightarrow 415$$

With the associated sequence governed by the rule, $u_i = u_{i-1} + u_{i-1} = 2u_{i-1}$ for $1 \leq i \leq 8$ and, $u_9 = u_8 + u_7$, and $u_{10+m} = u_{9+m} + u_{4-m}$, for $m = 0, 1, \ldots, 4$.

2.2 Artificial Immune System and Problem Representation

Our algorithm is based on a mechanism called clonal selection principle [10], that explains the way on which the antibodies eliminate a foreign antigen.

Fig. 1 depicts the clonal selection principle that establishes the idea that only those antibodies that best match the antigen are stimulated. The clonal selection principle achieved by means of two main mechanisms: antibody cloning and mutation. Stimulated antibodies are reproduced by cloning and the new clones suffer a mutation process with high rates (called hypermutation). The proliferation rate of each antibody is proportional to its affinity: the higher the affinity, the higher the number of offspring generated and vice versa. After this process is done, some of the newly created antibodies will increase their affinity to the antigens. Those clones will neutralize and eliminate the antigens. Once that the foreign antigens have been exterminated, the immune system must return to its normal values, eliminating the exceeding antibodies cells. However, the best cells remain into the body as memory cells.

According to de Castro and Timmis [11] in every artificial immune system, as in any other computational system with biological inspiration, the following elements must be defined:

A Representation of the System Components: A foreign antigen will be represented as the exponent e that we wish to reach. Antibodies will be

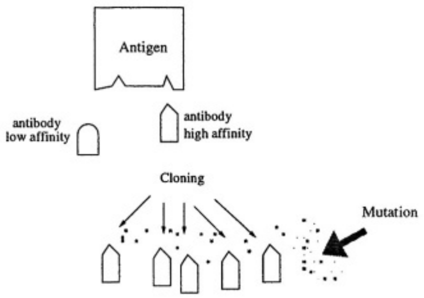


Fig. 1. The Clonal Selection Principle of the Immune System

represented by the addition chain sequence that contains the arithmetic recipe that we need to compute so that we can achieve the desired goal (the antigen). For instance, if we wish to reach the antigen $e = 415$ we can execute the antibody addition chain sequence: $1 \to 2 \to 3 \to 5 \to 10 \to 20 \to 40 \to 80 \to 90 \to 180 \to 360 \to 400 \to 410 \to 415$ which is a feasible problem solution (although not necessarily the best as we will see below).

Mechanisms to Evaluate the Interaction of Individuals with the Environment and Each Other: The fitness of a given antibody is determined by the length of its corresponding addition chain. The shorter the length is the better the associated fitness.

Procedures of Adaptation That Govern the Dynamics of the System: The dynamics of our system is based on the clonal selection principle.

2.3 AIS Algorithm

The AIS strategy to compute optimal addition chains is shown in Fig. 2. The exponent e is named the antigen or goal that the artificial immune system is trying to achieve. Starting with an initial population of three antibodies, the algorithm of Fig. 2 uses the cloning mechanism to generate slightly different replicas that are then selected based on the fitness of the individuals. A clone fitness is measured in terms of the length of its corresponding addition chain. In order to illustrate how our algorithm computes its task, let us consider the case when we want to obtain an optimal addition chain for the antigen $e = 415$.

Example 2. Given the antigen $e = 415$, then the algorithm of Fig. 2 performs as follows,

1. It starts with an initial population consisting of three antibodies Ab_1, Ab_2 and Ab_3 constructed according to the following rules,
   (a) Since the antigen e is an odd number, it proceeds directly to construct the three original antibodies.
   (b) Construction of the initial antibody Ab_1 using: 1 - 2 - 3
   (c) Construction of the initial antibody Ab_2 using: 1 - 2 - 4

```
Input: An exponent (antigen) $e$.
Output: An optimal addition chain (antibody) $U$.
Procedure AIS_Optimal_Addtion_Chain($e$, $U$)
1. Start with an initial population consisting of three antibodies $Ab_1, Ab_2$ and $Ab_3$
   constructed according to the following rules,
   1.a If the antigen $e$ is an even number of the form $e = 2^k e'$ then save $k$ and
   work with $e = e'$, with $e'$ an odd number.
   1.b Pre-initialize the antibody $Ab_1$ using the sequence: $1 \rightarrow 2 \rightarrow 3$
   1.c Pre-initialize the antibody $Ab_2$ using the sequence: $1 \rightarrow 2 \rightarrow 4$
   1.d Pre-initialize the antibody $Ab_3$ using the sequence: $1 \rightarrow 2 \rightarrow 3 \rightarrow 5$
   1.e Complete the addition chain sequence for the three germinal antibodies $Ab_1, Ab_2$
   and $Ab_3$ needed to achieve the antigen $e = e'$ trying to use the doubling rule,
   $u_i = u_{i-1} + u_{i-1} = 2u_{i-1}$ as much as possible.
   1.f Add to each antibody the $k$ doubling remaining steps needed to reach the original
   antigen $e$ of step 1.$a$. Now we have three valid addition chains associated to each one
   of the antibodies $Ab_1, Ab_2$ and $Ab_3$, that allow us to achieve the desired antigen $e$.
2. Repeat:
   3. Compute the associated fitness values of the three constructed antibodies, i.e.,
   the corresponding addition sequence lengths. Assign a better fitness to the antibodies
   with shorter lengths.
   4. Determine how many clones (copies) will be made per each antibody. The better
   the fitness the more clones to be made.
   5. Antibody Cloning: Apply the mutation operator to each clone as follows,
   5.a Randomly select a mutation point $i$ and a random number $j$ such that
   $0 \le j < i < e$
   5.b The new value of the clone's addition chain at the mutation point $u_i$ will be
   $u_i = u_{i-1} + u_j$
   5.c Update the upper part of the clone's addition chain so that the antigen $e$ is still
   reached by Applying as much doubling steps as possible.
   6. From the set of original antibodies and modified clones, select the top three and
   discard the rest. Those three survivor antibodies will make it to the next generation.
7. Go to step 2 a predetermined number of generations.
```

Fig. 2. Finding Optimal Addition Chains Using an AIS Strategy

(d) Construction of the initial antibody Ab_3 using: 1 - 2 - 3 - 5
(e) It proceeds to complete the antibodies. After performing as many doubling steps as possible we get,
Ab_1 : **1-2-3**-6-12-24-48-96-192-384-408-414-415
Ab_2 : **1-2-4**-8-16-32-64-128-256-384-400-408-412-414-415
Ab_3 : **1-2-3-5**-10-20-40-80-160-320-400-410-415

2. Repeat
3. Therefore, the corresponding fitness values (addition chain lengths) for each antibody are: $Ab_1 = 12$, $Ab_2 = 14$, $Ab_3 = 12$.
4. Since the fittest antibodies are Ab_1 and Ab_2, four clones of each one of them are made but only one clone of Ab_3.
5. **Antibody Cloning**: As an illustrative example let us analyze a clone *Cl* which is an exact replica of Ab_3 given as,
Cl : 1-2-3-5-10-20-40-80-160-320-400-410-415.
For this case the mutation mechanism will be applied as follows:
(a) A mutation point i and a random number j, say 160 and 10, respectively, are randomly selected.

(b) The number 160 will be changed by 90, which is obtained from the addition of 80+10. Therefore, the mutated (and mutilated) clone will look as, 1-2-3-5-10-20-40-80-**90.**
(c) The mutilated clone *Cl* must be restored by using as many doubling steps as possible until the antigen e is reached. Hence:
Cl:**1-2-3-5-10-20-40-80-90**-180-360-400-410-415.

6. From the whole population, the algorithm selects the best three solutions. They will be the antibodies for the next generation.
7. Go to step 2 a predetermined number of generations.

As it was explained, the above procedure creates nine clones at each iteration. After 100 generations our algorithm was able to find the following addition chain of length $l = 11$,

$$U : 1 \rightarrow 2 \rightarrow 3 \rightarrow 5 \rightarrow 10 \rightarrow 20 \rightarrow 40 \rightarrow 80 \rightarrow 83 \rightarrow 166 \rightarrow 332 \rightarrow 415 \quad (1)$$

The result obtained means a saving of 3 operations with respect to the binary method that, as it was seen in example 1, yields an addition chain of 14.

Proposition 1. *The shortest addition chain for* $e = 415$ *has length* 11.

Proof. After performing an exhaustive search, we found that there exist only two addition chains for $e = 415$ having a minimum length of 11, namely, the one described in Eq. 1 and its variant:

$$V : 1 \rightarrow 2 \rightarrow 3 \rightarrow 5 \rightarrow 10 \rightarrow 20 \rightarrow 40 \rightarrow 80 \rightarrow 85 \rightarrow 165 \rightarrow 330 \rightarrow 415$$

3 Multiplicative Inversion Over GF(2^m)

Since a quite long time it has been known that Fermat's Little Theorem (FLT) provides a mechanism to compute multiplicative inverses in finite fields [1]. Certainly, FLT establishes that for any nonzero element $\alpha \in GF(2^m)$, the identity $\alpha^{-1} \equiv \alpha^{2^m-2}$ holds. Therefore, multiplicative inversion can be performed by computing,

$$\alpha^{2^m-2} = \alpha^{2^1} \times \alpha^{2^1} \times \cdots \alpha^{2^{m-1}}. \quad (2)$$

A straightforward implementation of Eq. 2 can be carried out using the binary exponentiation method, which requires $m-1$ field squarings and $m-2$ field multiplications. The Itoh-Tsujii Multiplicative Inverse Algorithm (ITMIA) [8] on the other hand, reduces the required number of multiplications to $k+hw(m-1)-2$, where $k = \lfloor log_2(m-1) \rfloor$ and $hw(m-1)$ are the number of bits and the Hamming weight of the binary representation of $m-1$, respectively.

This remarkably saving on the number of multiplications is based on the observation that since $2^m - 2 = (2^{m-1}-1) \cdot 2$, then the identity from Fermat's little theorem can be rewritten as $\alpha^{-1} \equiv \alpha^{2^m-2} \equiv \alpha^{(2^{m-1}-1)^2}$. ITMIA computes the field element $(2^{m-1}-1)$ using a recursive re-arrangement of the finite field operations. It can be shown [8, 9] that the overall complexity of ITMIA is $m-1$

```
Input: An element $\alpha \in GF(2^m)$, an addition chain $U$ of length $l$ for $m-1$ and its
associated sequence V.
Output: $\alpha^{-1} \in GF(2^m)$
Procedure MultiplicativeInversion($\alpha$, $U$)
1. $\beta_0 = \alpha$
2. for i from 1 to $l$ do:
3.     $\beta_i = (\beta_{i_1})^{2^{u_{i_2}}} \cdot \beta_{i_2}$
4. return $(\beta_t{}^2)$;
```

Fig. 3. An Algorithm for Multiplicative Inversion using a Generalized Itoh-Tsujii Approach

field squarings plus $N = k + hw(m-1) - 2$ field multiplications, where $k = \lfloor log_2(m-1) \rfloor$ and $hw(m-1)$ are the number of bits and the Hamming weight of the binary representation of $m-1$, respectively.

The concept of addition chains leads us to a natural way to generalize the Itoh-Tsujii Algorithm. To see how this can be carried out let us first make some definitions.

Definition 1. *Let α be any arbitrary nonzero element in the field $GF(2^m)$. Then we define $\beta_k \in GF(2^m), k$ a positive integer, as*

$$\beta_k = \alpha^{2^k - 1} \tag{3}$$

Lemma 1. *Let k, j be two positive integers. Then, the element $\beta_{k+j} \in GF(2^m)$ can be expressed as,*

$$\beta_{k+j} = \beta_k{}^{2^j} \cdot \beta_j = \beta_j{}^{2^k} \cdot \beta_k \tag{4}$$

Proof. Using Definition 1 to substitute β_k and β_j in terms of the variable α we obtain,

$$\begin{aligned} \beta_k{}^{2^j} \cdot \beta_j &= (\alpha^{2^k-1})^{2^j} \cdot \alpha^{2^j-1} \\ &= \alpha^{2^{k+j}-2^j} \cdot \alpha^{2^j-1} \\ &= \alpha^{2^{k+j}-2^j+2^j-1} \\ &= \alpha^{2^{k+j}-1} = \beta_{k+j} \end{aligned}$$

□

Consider now the algorithm shown in Fig. 3. That algorithm iteratively computes the β_i coefficients in the exact order stipulated by the addition chain *U*. Indeed, starting from $\beta_0 = (\alpha)^{2^{u_0}-1} = (\alpha)^{2^{u_0}-1} = \alpha^{2^1-1}$, the algorithm computes the other l β_i coefficients. In the final iteration, after having computed the coefficient $\beta_l = (\alpha)^{2^{m-1}-1}$, the algorithm returns the required multiplicative inversion by performing a regular field squaring, namely, $\beta_l^2 = (\alpha^{2^m-2}) = \alpha^{-1}$.

3.1 A Design Example

Let $\alpha \in GF(2^{415})$ be an arbitrary nonzero field element. Then, using the addition chain of Example 2, the algorithm of Fig. 3 will compute the sequence of β coefficients as shown in Table 1. Once again, notice that after having computed the coefficient β_{11} the only remaining step is to obtain α^{-1} as, $\alpha^{-1} = \beta_{11}^2$.

Table 1. Algorithm of Fig. 3: β_i Coefficient Generation

i	u_i	rule	$\beta_{i_1}^{2^{u_{i_2}}} \cdot \beta_{i_2}$	$\beta_i = \alpha^{2^{u_i}-1}$
0	1	–	–	$\beta_0 = \alpha^{2^1-1}$
1	2	$2u_{i-1}$	$\beta_0^{2^1} \cdot \beta_0$	$\beta_1 = \alpha^{2^2-1}$
2	3	$u_{i-1} + u_{i-2}$	$\beta_1^{2^1} \cdot \beta_0$	$\beta_2 = \alpha^{2^3-1}$
3	5	$u_{i-1} + u_{i-2}$	$\beta_2^{2^2} \cdot \beta_1$	$\beta_3 = \alpha^{2^5-1}$
4	10	$2u_{i-1}$	$\beta_3^{2^5} \cdot \beta_3$	$\beta_4 = \alpha^{2^{10}-1}$
5	20	$2u_{i-1}$	$\beta_4^{2^{10}} \cdot \beta_4$	$\beta_5 = \alpha^{2^{20}-1}$
6	40	$2u_{i-1}$	$\beta_5^{2^{20}} \cdot \beta_5$	$\beta_6 = \alpha^{2^{40}-1}$
7	80	$2u_{i-1}$	$\beta_6^{2^{40}} \cdot \beta_6$	$\beta_7 = \alpha^{2^{80}-1}$
8	83	$u_{i-1} + u_2$	$\beta_7^{2^3} \cdot \beta_2$	$\beta_8 = \alpha^{2^{83}-1}$
9	166	$2u_{i-1}$	$\beta_8^{2^{83}} \cdot \beta_8$	$\beta_9 = \alpha^{2^{166}-1}$
10	332	$2u_{i-1}$	$\beta_9^{2^{166}} \cdot \beta_9$	$\beta_{10} = \alpha^{2^{332}-1}$
11	415	$u_{i-1} + u_{i-3}$	$\beta_{10}^{2^{166}} \cdot \beta_9$	$\beta_{11} = \alpha^{2^{415}-1}$

Table 2. Optimal Addition Chains for $m = 32k$. AIS=Artificial Immune System

m	$m-1$	AIS	AIS	Takagi [9]	ITMIA Binary method [8]
32	31	1 2 3 6 7 14 28 31	7	7	8
64	63	1 2 3 6 7 14 28 56 63	8	8	10
96	95	1 2 3 5 10 20 40 80 90 95	9	9	11
128	127	1 2 3 6 12 24 48 96 120 126 127	10	10	12
160	159	1 2 3 6 12 24 48 96 144 156 159	10	10	12
192	191	1 2 4 8 16 17 34 68 136 170 187 191	11	11	13
224	223	1 2 3 6 12 13 26 52 104 208 221 223	11	11	13
256	255	1-2-3-5-10-20-40-80-85-170-255-	10	10	14
288	287	1-2-3-5-7-14-28-56-112-224-280-287-	11	11	13
320	319	1-2-3-6-12-18-36-72-144-288-306-318-319-	12	12	14
352	351	1-2-3-6-12-24-27-54-108-216-324-351-	11	11	14
384	383	1-2-3-5-10-20-40-80-160-320-360-380-383-	12	13	15
416	415	1-2-3-5-10-20-40-80-83-166-332-415-	11	12	14
448	447	1-2-3-6-12-18-36-72-144-288-432-444-447-	12	12	15
480	479	1-2-3-6-7-14-28-56-112-224-448-476-479-	12	13	15
512	511	1-2-3-5-10-15-30-60-120-240-480-510-511-	12	12	16
576	575	1-2-3-5-10-20-23-46-92-184-368-552-575-	12	13	15
608	607	1-2-3-6-12-18-36-72-144-288-576-594-606-607-	13	13	15
640	639	1-2-3-6-12-24-26-52-104-208-416-624-636-639-	13	13	16
704	703	1-2-3-5-10-20-40-80-160-320-640-680-700-703-	13	13	16
736	735	1-2-3-5-10-15-30-60-120-240-480-720-735-	12	12	16
768	767	1-2-3-5-10-20-40-80-83-166-332-664-747-767-	13	14	17
800	799	1-2-3-6-12-24-48-96-192-384-768-792-798-799-	13	13	15
832	831	1-2-3-6-12-24-48-96-192-384-768-816-828-831-	13	13	16
864	863	1-2-3-5-10-20-40-43-86-172-344-688-860-863-	13	15	16
896	895	1-2-3-5-10-20-40-80-160-163-326-652-815-895-	13	14	17

4 Result Comparison

We compare the results obtained by our algorithm against the modified factor method presented by Takagi et al [9] (whose results are the best known to date) and the ITMIA binary method [8]. Table 2 shows the optimal addition chains for m=32k (which is an important exponent form for error-correcting code applications). The chains found by the AIS algorithm are summarized in the second and third columns, where all of them were obtained after 100 generations. On a total of seven cases the AIS algorithm presented here outperforms the method of [9]. And in all the cases, those two algorithms outperform the ITMIA binary method.

Table 3 summarizes the results obtained by AIS and the binary method for $e = p$ number, with p a prime number (which is an important exponent form for elliptic curve cryptography). In most of the cases, the immune algorithm obtains better results than the ITMIA binary method.

Table 3. Optimal Addition Chains for $e = p - 1$, p a prime

p	$p-1$	AIS	AIS	ITMIA Binary method [8]
163	162	1 2 4 8 16 32 64 80 81 162	9	9
167	166	1 2 3 5 10 20 40 80 83 166	9	10
173	172	1 2 3 5 10 20 40 43 86 172	9	10
191	190	1 2 3 5 10 20 40 80 160 180 190	10	12
193	192	1 2 3 6 12 24 48 96 192	8	8
197	196	1 2 4 8 16 32 48 49 98 196	9	9
223	222	1 2 3 6 12 24 48 96 192 216 222	10	12
2 233	232	1 2 4 8 16 24 28 29 58 116 232	10	10
269	268	1 2 4 8 16 32 64 66 67 134 268	10	10
271	270	1 2 3 5 10 20 40 80 90 180 270	10	11
293	292	1 2 4 8 16 32 64 72 73 146 292	10	10
331	330	1 2 3 5 10 20 40 80 160 320 330	10	11
379	378	1 2 4 8 16 18 36 72 144 288 360 378	11	13
383	382	1 2 3 5 10 20 40 80 160 320 360 380 382	12	14
389	388	1 2 4 8 16 32 64 96 97 194 388	10	10
443	442	1 2 3 6 12 13 26 52 104 208 416 442	11	13
463	462	1 2 4 8 16 32 33 66 132 264 396 462	11	13
491	490	1 2 3 5 10 20 40 80 160 320 480 490	11	13
509	508	1 2 3 6 12 14 28 30 60 120 240 480 508	12	14
521	520	1 2 4 8 16 32 64 65 130 260 520	10	10

5 Conclusions

In this paper we described how an artificial immune system can be applied to the problem of finding optimal addition chains for field exponentiation computations. The results obtained by our scheme yield the shortest reported lengths for exponents typically used when computing field multiplicative inverses for error-correcting and elliptic curve cryptographic applications. Future work includes finding addition chains for bigger exponents such as the ones typically used in RSA and DSA cryptosystems. We would like also to explore the feasibility of applying other biological-inspired heuristics to the optimal addition chain problem.

References

1. Knuth, D.E.: The Art of Computer Programming 3rd. ed. Addison-Wesley, Reading, Massachusetts (1997)
2. Menezes, A.J., van Oorschot, P.C., A.Vanstone, S.: Handbook of Applied Cryptography. CRC Press, Boca Raton, Florida (1996)
3. Gordon, D.M.: A survey of fast exponentiation methods. Journal of Algorithms **27** (1998) 129–146
4. Lamont, G.B., Marmelstein, R.E., Veldhuizen, D.A.V.: A distributed architechture for a self-adaptive computer virus immune system. In: New Ideas in Optimization. Mc Graw-Hill (1999) 167–183
5. Forrest, S., Hofmeyr, S.A.: Immunology as Information Processing. In Segel, L., Cohen, I., eds.: Design Principles for the Immune System and Other Distributed Autonomous Systems. Santa Fe Institute Studies in the Sciences of Complexity. Oxford University Press (2000) 361–387
6. Coello Coello, C.A., Cruz-Cortés, N.: A parallel implementation of an artificial immune system to handle constraints in genetic algorithms: Preliminary results. In: Proceedings of the special sessions on artificial immune systems in the 2002 Congress on Evolutionary Computation, 2002 IEEE World Congress on Computational Intelligence, Honolulu, Hawaii (2002)
7. Toma, N., Endo, S., Yamada, K.: Immune Algorithm with Immune Network and MHC for Adaptive Problem Solving. In: Proceedings of the IEEE System, Man, and Cybernetics. Volume IV. (1999) 271–276
8. Itoh, T., Tsujii, S.: A fast algorithm for computing multiplicative inverses in $\mathrm{GF}(2^m)$ using normal basis. Information and Computing **78** (1988) 171–177
9. Takagi, N., Yoshiki, J., Tagaki, K.: A fast algorithm for multiplicative inversion in $\mathrm{GF}(2^m)$ using normal basis. IEEE Transactions on Computers **50(5)** (2001) 394–398
10. Burnet, F.M.: Clonal selection and after. In Bell, G.I., Perelson, A.S., Jr., G.H.P., eds.: Theoretical Immunology, Marcel Dekker Inc. (1978) 63–85
11. de Castro, L., Timmis, J.: An Introduction to Artificial Immune Systems: A New Computational Intelligence Paradigm. Springer-Verlag (2002)

Particle Swarm Optimization in Non-stationary Environments

Susana C. Esquivel[1] and Carlos A. Coello Coello[2]

[1] Laboratorio de Investigación y Desarrollo en Inteligencia Computational, Universidad Nacional de San Luis, Ejército de los Andes 950, (5700) San Luis, Argentina
esquivel@unsl.edu.ar

[2] CINVESTAV-IPN (Evolutionary Computation Group), Electrical Engineering Department, Computer Science Section, Av. IPN No. 2508, Col. San Pedro Zacatenco, México D.F. 07300, México
ccoello@cs.cinvestav.mx

Abstract. In this paper, we study the use of particle swarm optimization (PSO) for a class of non-stationary environments. The dynamic problems studied in this work are restricted to one of the possible types of changes that can be produced over the fitness landscape. We propose a hybrid PSO approach (called HPSO_dyn), which uses a dynamic macromutation operator whose aim is to maintain diversity. In order to validate our proposed approach, we adopted the test case generator proposed by Morrison & De Jong [1], which allows the creation of different types of dynamic environments with a varying degree of complexity. The main goal of this research was to determine the advantages and disadvantages of using PSO in non-stationary environments. As part of our study, we were interested in analyzing the ability of PSO for tracking an optimum that changes its location over time, as well as the behavior of the algorithm in the presence of high dimensionality and multimodality.

1 Introduction

Many real-world optimization problems are non-stationary. Such problems arise when either the resources (constraints) or the optimization criteria change (or both), and they are common in engineering, production planning and economics [2]. In the last decade, heuristics that can adapt to changes have attracted a lot of interest, particularly those that operate on a population of solutions. From these heuristics, most of the research has concentrated on evolutionary algorithms [3, 4], and several promising results have been obtained within the last few years. Another population-based technique that has recently been adopted for dealing with non-stationary environments is particle swarm optimization [5, 6], which is precisely the approach adopted in the work reported in this paper. An algorithm capable of dealing with non-stationary environments normally considers two main aspects [4]: the detection of the change and the reaction of the algorithm to such change, such that the (moving) optimum can be tracked as closely as possible. There is some previous research on the use of particle swarm optimization for dynamic optimization that involves these two aspects previously discussed. Carlisle & Dozier [7] used a *sentry particle* that is evaluated at each iteration, comparing the previous fitness value with the new one (if they differ, this means that the environment has changed). Hu & Eberhart [8] proposed the re-evaluation of two g_{best} particles in order to detect the movement of the location of the optimum. Several authors have also proposed approaches in which the main emphasis is the reaction of the algorithm to changes in the environment (see for example [9, 10, 11]). In this paper, we will focus on this second aspect. Despite the encouraging results reported in the papers previously indicated, all of these authors focus their work on unimodal fitness landscapes [10, 11]. This is rather unrealistic if we consider the complexity of real-world problems that frequently present high multimodality. This is precisely the issue that we address in this paper. Aiming to provide a framework for performing further comparative studies, we decided to adopt the test case generator originally developed by Morrison & De Jong [1], which shares several similarities with the proposal of Branke [4]. This test case generators allows us to define non-stationary environments of different degrees of complexity both regarding the morphology of the fitness landscapes as well as regarding the type of changes that can be produced and their severity.

2 Classical PSO Model

The PSO model (now considered "classical") was originally proposed by James Kennedy and Russell Eberhart in 1995 [12], and has had several further extensions in the next few years [5]. The PSO model is inspired on the acting of communities that present both an individual and a social behavior, from which the main socio-cognitive ideas may be applied to the development of efficient algorithms that solve optimization problems (this is the so-called "swarm intelligence" [5]). Although PSO is considered by its authors as an evolutionary algorithm, it does not incorporate the traditional genetic operators (crossover and mutation) that evolutionary algorithms normally adopt. Instead, in PSO each particle adjusts its flight over the search space based on its own flight experience and that of its neighbors (i.e., the other particles of its swarm). Each particle represents a possible solution to the problem at hand and is treated as a point in a d-dimensional search space. A particle is characterized by its position and its current velocity. Furthermore, a particle has a memory where it stores the best position that it has found so far. Particles adjust their flight trajectories using the following equations [13]:

$$v_{i,j} = w \times v_{i,j} + c_1 \times r_1 \times (p_{i,j} - x_{i,j}) + c_2 \times r_2 \times (p_{g,j} - x_{i,j}) \tag{1}$$

$$x_{i,j} = x_{i,j} + v_{i,j} \tag{2}$$

where w is the inertia factor, whose goal is to control the impact of the history of the velocities of a particle over the current velocity, by influencing the local and global exploration abilities of the algorithm. $v_{i,j}$ is the velocity of the particle i in the j-th dimension, c_1 and c_2 are weights applied to the influence of the best position found so far by particle i and by the best particle in the swarm g. r_1 and r_2 are random values (with a uniform distribution) within the interval [0,1]. After the velocity is updated, the new position of the particle i in its jth dimension is recomputed. This process is repeated for each dimension of the particle i and for all the particles in the swarm. Kennedy and Eberhart [14] proposed different neighborhood techniques to be considered when updating the velocity of a particle. In this paper, we adopted the most simple neighborhood scheme available, which only includes the closest neighbors in terms of the indices of the particles. Under this scheme, the equation to update the velocity of a particle becomes:

$$v_{i,j} = w \times v_{i,j} + c_1 \times r_1 \times (p_{i,j} - x_{i,j}) + c_2 \times r_2 \times (p_{l,j} - x_{i,j}) \quad (3)$$

where $p_{l,j}$ represents to the best particle in the neighborhood. The formula used to update a particle remains the same as before.

3 Hybrid PSO Model

Recently, there have been several proposals of PSO approaches hybridized with evolutionary algorithms' concepts and operators [15, 16]. The proposal presented in this paper has as its basis the equations to update the particles and velocities defined by equations (2) and (3). Additionally, we use a dynamic macromutation operator that is described next.

Mutation: Each coordinate of the particle is independently mutated with a probability p_{mut}. The coordinate value is replaced by another value which is randomly generated within the allowable range. The mutation probability is dynamically adjusted taking values within the interval $[p_{min}, p_{max}]$ during the execution of the algorithm. Thus, p_{mut} is defined as follows:

$$p_{mut} = \frac{(p_{max} - p_{min}) \times (interval - iter_{current} \bmod interval)}{interval} + p_{min} \quad (4)$$

where the values p_{min} and p_{max} are the lower and upper bounds of the variation interval of the probability, *interval* indicates the number of iterations between changes and $iter_{current}$ corresponds to the current flight cycle of the particles. Each time a change takes place in the environment, the value of p_{mut} is initialized to p_{max} and the end of the interval of p_{mut} is set to p_{min}.

Once the swarm and the velocities are initialized (lines 1–2), the swarm is evaluated with the base function F_0. Then, the memory of the particles *(Swarm_bests)* is initialized (line 4). After that, the algorithm enters the flight loop (line 6) which is executed during a number of cycles that is determined in terms of the number of changes to be performed and in terms of the interval between such changes. Then, on line 7, we compute the value for p_{mut} (according to equation (4)) and the function `occurred_changes` determines if a change is required within the current flight cycle. In order to do this, we check if the current cycle number is a multiple of the number of cycles between changes. The changes in the environment are produced at constant intervals. If the environment must change, the dynamical statistics are reported, the function is modified and both the *Swarm* and the *Swarm_bests* are re-evaluated with the new fitness function (lines 9 to 12). These re-evaluations are necessary since the current fitnesses of the particles do not correspond to the new function. Once the two swarms have been re-evaluated, we

```
1. Initialize($Swarm$)
2. Initialize($velocities$)
3. Evaluate($Swarm, F_0$)
4. Copy($Swarm, Swarm\_bests$)
5. $t = 0$
6. do
7.     Calculate($p_{mut}$)
8.     if ($occurred\_change$)
9.         Report_dynamic_statistics()
10.        Change_function()
11.        Evaluate($Swarm, F_t$)
12.        Evaluate($Swarm\_bests, F_t$)
13.        Update($Swarm\_bests$) if appropriate
14.        Calculate($p_{mut}$)
15.    end if
16.    Mutate($Swarm$)
17.    Update($velocities$)
18.    Update($Swarm$)
19.    Evaluate($Swarm, F_t$)
20.    Update($Swarm\_bests$) if appropriate
21.    $t = t + 1$
22. while ($\neg termination$)
```

Fig. 1. General outline of the HPSO_dyn Algorithm

verify, for each particle, that *Swarm_bests* really contains the best particle positions for the new environment, as to maintain the consistency of the algorithm. By doing this, we do not lose the memory of all the particles, but only the memory of those for which their current positions (in *Swarm*) are the best [11]. We recompute again the transition function for the probability of mutation that will take us from p_{mut} to p_{max}. Next, we proceed with the normal PSO processing, which implies: mutate the particles, update their velocities, update the positions of the particles, evaluate the particles and, if applicable, modify their position in *Swarm_bests* (lines 16–20). This process is done asynchronously, since there is evidence of the efficiency of this type of processing when working with neighborhoods in PSO [16, 6].

4 DF1 Generator

In this section we briefly describe the Test Function Generator proposed by Morrison & De Jong [1]. This generator uses a morphology that the authors denominate "field of cones". Such cones can have different heights and slopes and are randomly distributed along the landscape. The static base function (F_0), for the two-dimensional case, is defined as:

$$F_0(x, y) = \max_{i=1,n}[H_i - R_i \times \sqrt{(x - x_i)^2 + (y - y_i)^2}] \qquad (5)$$

where n indicates the number of cones in the landscape and each cone is independently specified by its coordinates (x_i, y_i) that belong to the interval [–1, 1], its height (H_i) and its slope (R_i). The cones independently defined are grouped using the function *max*. Each time the generator is invoked, it randomly generates a morphology with the characteristics given. The function F can be defined for any number of dimensions. The user can create a wide variety of shapes for the fitness landscape by specifying the range of allowable values for the height, slope and location of the cones. Furthermore, the generator allows to re-write the values randomly generated in order to create landscapes with controlled characteristics. The severity of the changes is controlled through the *logistic function:*

$$Y_i = A \times Y_{i-1} \times (1 - Y_{i-1}) \tag{6}$$

where A is a constant defined within the interval $[1, 4]$ and Y_i is the value at the iteration i. Thus, the procedure to obtain the dynamic that the user wants is the following: 1) choose the number of cones and 2) determine what characteristics the user wishes to change (height, location, slope of one or all the cones). The severity of the change is controlled by the values that are assigned to the constant A, since from this depends that the movement is produced either at small, large or chaotic steps. For further details on this generator, the reader should refer to [1].

5 Experimental Design

The goal of this research was twofold: first, we wanted to determine if the proposed algorithm could track down the optimum once a change has occurred. Second, we wanted to analyze the algorithm's behavior upon scaling both the number of dimensions and the number of cones. Thus, we considered that the scenarios proposed by Morrison [17] provided the required conditions to perform our study. Such scenarios are described next: 1) by the shapes of the fitness landscape and 2) by the dynamics applied.

Description of the structure of the static landscape

1. E1: 2 dimensions, 5 cones (2d-5c).
2. E2: 2 dimensions, 14 cones (2d-14c).
3. E3: 5 dimensions, 5 cones (5d-5c).
4. E4: 10 dimensions, 14 cones (10d-14c).

In all the scenarios, we assigned the maximum value to a single peak. Such a value is greater than that of the other peaks (to make sure that there is a single global optimum).

Dynamics Applied
The type of change implemented consisted of modifying only the location of the cone that contains the optimum (CO) or changing the location of all the cones simultaneously (TC). In Table 1, we describe, for each problem studied, the type of change and its severity.

For the scenarios $E1$ to $E3$, the changes take place at every 10, 20, 30, 40 and 50 iterations. For scenario $E4$, given its complexity, changes take place at every 30, 60 and

Table 1. Dynamic Behavior Applied

Scenarios	Dimension	Cones	Change_type	Step_size
E1	2	5	CO	small(s)
E2	2	14	TC	chaotic(c)
E3	5	5	TC	small
E3	5	5	TC	large(l)
E4	10	14	TC	chaotic

90 iterations. Furthermore, at each run, 20 changes took place either on the location of the cone that contains the optimum value or in all the cones.

Parameters ofthe DF1 Generator
The parameters of DF1 that correspond to the dynamics implemented are defined in Table 2, according to the proposal by [17], where *Hbase, Hrange, Rbase* and *Rrange* correspond to the ranges of the heights and slopes of the cones that do not contain the global optimum. *OptH* and *OptR* correspond to the cone containing the global optimum.

Table 2. DF1 Parameters

Par	2d-5c-CO-s	2d-14c-TC-c	5d-5c-TC-s	5d-5c-TC-l	10d-14c-TC-c
Hbase	60.0	1.0	60.0	60.0	1.0
Hrange	0.0	9.0	0.0	0.0	9.0
Rbase	70.0	8.0	70.0	70.0	8.0
Rrange	0.0	12.0	0.0	0.0	12.0
Ac	1.5	3.8	1.5	1.5	3.8
Scale	0.3	0.5	0.3	0.99	0.5
OptH	90.0	15.0	90.0	90.0	15.0
OptR	90.0	20.0	90.0	90.0	20.0

Parameters of the HPSO_dyn Algorithm
The parameters required by the algorithm are provided in Table 3 and were selected after a large series of experiments.

Table 3. PSO Algorithm Parameter Settings

Scenarios	w	$c1$	$c2$	p_{min}	p_{max}	$swarm_{size}$
E1	0.5	1.5	1.5	0.1	0.4	50
E2	0.5	1.5	1.5	0.5	0.8	50
E3	0.5	1.5	1.5	0.3	0.6	200
E4	0.5	1.5	1.5	0.5	0.8	500

In all the experiments, we worked with a neighborhood radius of size 4. For each experiment, we performed 30 runs, all of them with the same base function and the same initial population.

6 Analysis of Results

In Table 4 we show the results obtained for the 2D (i.e., two-dimensional) functions with 5 and 14 cones. Here, $\overline{f}$ refers to the mean of the best average fitness values found in the 30 runs and s is the variance. The optimum value for all the functions with 5 cones is 90.00 and all the other cones have a maximum value of 60.00. For the functions with 14 cones, the maximum value is 15.00 and all the other cones have a maximum of 10.00. The performance of our HPSO_dyn for the two-dimensional functions with 5 and 14 cones is very satisfactory, since in all cases, for all the changes performed, the algorithm was able to track down the global optimum with an acceptable error.

Table 4. Two-dimensional functions

Int_{chgs}	2d-5c-CO-s	2d-14c-TC-s
10	600 (100%)	600 (100%)
$\overline{f}$	89.695084	14.848885
s	0.042112	0.027802
20	600 (100%)	600 (100%)
$\overline{f}$	89.966666	14.983450
s	0.274450	0.003067
30	600 (100%)	600 (100%)
$\overline{f}$	89.997041	14.998491
s	0.02776	0.002023
40	600 (100%)	600 (100%)
$\overline{f}$	89.999650	14.999882
s	0.000252	0.000354
50	600 (100%)	600 (100%)
$\overline{f}$	89.999973	14.999358
s	0.000032	0.000177

Table 5 shows the results obtained by our algorithm for the 5D functions with 5 cones, in the case where all the cones change their location in the landscape, with step sizes which are, in one case small, and in the other one, large. In this case, our algorithm is able to track down the optimum upon performing all the changes. However, note that the error becomes higher when the changes are produced at every 10 iterations. Finally, Table 6 presents the results obtained for the most complex environment studied in this paper: a 10D function with 14 cones, with changes produced with a chaotic step size. As can be observed, the performance of the algorithm was degraded since we could not succeed 100% of the time at tracking down the optimum in any of our experiments. Nevertheless, the success rate oscillates between 87% and 93%, which are values reasonably good. It is also worth noticing that the difference between the fitness values found by the algorithm with respect to the global optimum increased as well. This is because we included in the average fitness values the unsuccessful cases. Table 7 provides the success rate of our algorithm for the case of 10D functions with 14 cones. This table indicates (as a percentage) the number of runs in which the algorithm was able to successfully track down the global optimum (inputs in the tables). Note that 20 changes took place per run.

Table 5. Functions of 5 dimensions with 5 cones

Int_{chgs}	5d-5c-TC-l	5d-5c-TC-s
10	600 (100%)	600 (100%)
$\bar{f}$	79.618352	78.821386
s	0.913599	0.934210
20	600 (100%)	600 (100%)
$\bar{f}$	86.373647	85.169944
s	0.997890	1.003454
30	600 (100%)	600 (100%)
$\bar{f}$	88.554862	88.960822
s	1,137792	1.20767
40	600 (100%)	600 (100%)
$\bar{f}$	89.606563	89.673193
s	0.061213	0.077778
50	600 (100%)	600 (100%)
$\bar{f}$	89.865952	89.849669
s	0.029567	0.013788

Table 6. 10d-14c Function with chaotic changes for all cones

Int_{chgs}	10d-14c-TC-c
30	527 (87,8%)
$\bar{f}$	11.813951
s	0.181945
60	550 (91,6%)
$\bar{f}$	13.74900
s	0.111303
90	560 (93,3%)
$\bar{f}$	13.983545
s	0.201526

Table 7. Success Rates for the Function 10d-14c

%	30	60	90
100	-	-	-
95	4	10	20
90	13	20	10
85	10	-	-
80	3	-	-

As we can see in the table even in the most unfavorable case, when the interval between changes is small (30 iterations) the algorithm is able to adapt between the 85% and 90% of the changes. As we expected, these percentages variate between 90% and 95% when the range of the interval is increased.

7 Conclusions and Future Work

We have presented an optimization algorithm for non-stationary environments. Our algorithm is based on a particle swarm optimizer which was hybridized with a dynamic macromutation operator. Although one could think at first sight that the mutation probabilities adopted in our work are too high, this is done to maitain the required diversity in the swarm and does not disrupt the search process, because of the use of the memory *Swarm_bests,* which is not destroyed when the changes take place and is only updated when a particle from the *Swarm* obtains a better fitness than the one stored in such memory. In our experiments, we adopted functions with 2, 5 and 10 dimensions and with 5 and 14 cones. All of these functions were created with the DF1 generator, which allowed us to study the behavior of the algorithm with more complex fitness landscape structures than those normally adopted in the specialized literature (i.e., the sphere model). The changes produced consisted either on changing the location of the cone that contained the global optimum or on changing the location of all the cones, with step sizes that were from small to chaotic. The results obtained show that the performance of the algorithm over morphologies with several suboptima, is highly satisfactory when using either 2 or 5 dimensions and a number of cones between 5 and 14. However, the performance degrades as we move to problems with 10 dimensions. Nevertheless, the success rate of the algorithm in these cases remains reasonably good, since it only fails to track down an average of 2 (out of 20) changes performed. Although the scenarios presented in this paper were non-trivial, our future work adresses two aspects: 1) to extend these scenarios to include other types of landscape morphologies and 2) to study the other change types provided by the DF1 generator.

Acknowledgments

The first author acknowledges the continuous support received from the Universidad Nacional de San Luis and the ANPCYT. The second author acknowledges support from CONACyT project number 42435-Y.

References

1. Morrison, R.W., De Jong, K. A.: A Test Problem Generator for Non-Stationary Environments. In: Conference on Evolutionary Computation (CEC'99). Volume 3., Piscataway, NJ., IEEE Service Center (1999) 2047–2053
2. Michalewicz, Z., Fogel, D.B.: How to Solve it: Modern Heuristics. Springer, Berlin (2000)
3. Angeline, P.J.: Tracking Extrema in Dynamic Environments. In et al., P.J.A., ed.: Evolutionary Programming VI, 6th International Conference EP'97, Springer-Verlag. Lecture Notes in Computer Science No. 1213 (1997) 335–345
4. Branke, J.: Evolutionary Optimization in Dynamic Environments. Kluwer Academic Publishers, Boston/Dordecht/London (2002)
5. Kennedy, J., Eberhart, R.C.: Swarm Intelligence. Morgan Kaufmann Publishers, California, USA (2001)
6. Carlisle, A.: Applying The Particle Swarm Optimization to Non-Stationary Environments. PhD thesis, Auburn University, USA (2002)

7. Carlisle, A., Dozier, G.: Adapting Particle Swarm Optimization to Dynamic Environments. In: International Conference on Artificial Intelligence, Las Vegas, Nevada (2000) 429–434
8. Hu, X., Eberhart, R.C.: Adaptive particle swarm optimization: Detection and response to dynamic systems. In: International Conference on Evolutionary Algorithms, Piscataway, NJ., IEEE Press (2002)
9. Hu, X., Eberhart, R.: Tracking Dynamic Systems with PSO: Where's the Cheese? In: Workshop on Particle Swarm Optimization, Purdue school of engineering and technology (2001)
10. Blackwell, T.: Swarms in Dynamic Environments. In et al., E.C.P., ed.: Genetic and Evolutionary Computation—GECCO 2003. Proceedings, Part I, Springer. Lecture Notes in Computer Science Vol. 2723 (2003) 1–12
11. Carlisle, A., Dozier, G.: Tracking Changing Extrema with Adaptive Particle Swarm Optimizer. In: World Automation Congress, Orlando, USA (2002)
12. Kennedy, J., Eberhart, R.C.: Particle Swarm Optimization. In: Proceedings of the 1995 IEEE International Conference on Neural Networks, Piscataway, NJ., IEEE Service Center (1995) 1942–1948
13. Shi, Y., Eberhart, R.: A Modified Particle Swarm Optimizer. In: Proceedings of the 1998 IEEE International Conference on Evolutionary Computation, Piscataway, NY., IEEE Service Center (1998) 69–73
14. Kennedy, J., Mendes, R.: Population Structure and Particle Swarm Performance. In: Congress on Evolutionary Computation (CEC'2002). Volume 2., Piscataway, New Jersey, IEEE Service Center (2002) 1671–1676
15. Angeline, P.J.: Using selection to improve particle swarm optimization. In: Proceedings of the 1998 IEEE International Conference on Evolutionary Computation, Piscataway, NJ., IEEE Service Center (1998) 84–89
16. Esquivel, S., Coello Coello, C.A.: On the Use of Particle Swarm Optimization with Multimodal Functions. In: 2003 Congress on Evolutionary Computation (CEC'2003). Volume 2., IEEE Press (2003) 1130–1136
17. Morrison, R.W.: Designing Evolutionary Algorithms for Dynamic Environments. PhD thesis, George Mason University, Fairfax, Virginia, USA (2002)

An Efficient Learning Algorithm for Feedforward Neural Network

Songbo Tan[1,2] and Jun Gu[3]

[1] Software Department, Inst. of Computing Tech., CAS, P.O. Box 2704, Beijing, 100080, P.R. China
[2] Graduate School of the Chinese Academy of Sciences, P.R. China
tansongbo@software.ict.ac.cn
[3] Department of Computer Science, Hong Kong University of Science and Technology
eecs@263.net

Abstract. BP algorithm is frequently applied to train feedforward neural network, but it often suffers from slowness of convergence speed. In this paper, an efficient learning algorithm and its improved algorithm based on local search are proposed. Computer simulations with standard problems such as XOR, Parity, TwoNorm and MushRoom problems are presented and compared with BP algorithm. Experimental results indicate that our proposed algorithms achieve accuracy much better than BP algorithm and convergence speed much faster than BP algorithm, and the generalization of our proposed algorithms for TwoNorm and MushRoom problems is comparable to BP algorithm.

1 Introduction

Since the presentation of the MP model [1] in 1943, neural network has received a growing research interest. The feedforward neural network is one of the most important and popular neural networks. Back-propagation (BP) algorithm is one of the most popular training algorithms for feedforward neural network. Unfortunately it is very slow for practical applications and its training performance is sensitive to the choice of learning rate and initial values of the weight parameters.

Given a neural network and a set of training examples, Judd has proved that the search of a set of weights for the network so that the network produces correct output for all training examples is NP-completeness [2]. Even if the network is required to produce correct output for only two-thirds of the training examples, the problem still is NP-completeness. Blum also has proved that training 3-nodes network so that it can produce correct output is NP-completeness [3]. Therefore, we can say, the training of neural network is intrinsically difficult.

Local search was one of the earliest approaches proposed to cope with the computational intractability of NP-hard combinatorial optimization problems. It has made great success in NP-hard problems in recent years [4]-[9]. Consequently we introduce local search to train feedforward neural network. We test our algorithm in many problems varying from small training sets to large training sets. The results of tests indicate that our proposed algorithms outmatch the BP algorithm and its improved versions with momentum.

The rest of this paper is organized as follows. In section 2, we briefly review prior work. Section 3 introduces local search. The proposed algorithms are presented in section 4. Performance results are shown in section 5. Section 6 concludes this paper.

2 Prior Work

Due to the time required to train a Neural Network and the sensitivity to initial weights, many researchers have devoted their efforts to developing speedup techniques. Various efforts range from optimizations of current algorithms to development of original algorithms. These modified and novel algorithms can be divided into three parts: the modifications of BP algorithm, new algorithms based on different mathematical optimization methods and algorithms based on stochastic search methods.

The first kind of algorithms are mainly involved in the modifications of the learning rate. Chan [10] suggested a new method to adapt a global learning rate during the training. The method adjusts the learning rate by observing the angle between $\nabla E(n)$ and $\Delta W(n-1)$, and tries to adjust this angle at 90°. As long as the angle is less than 90 the learning rate is increased otherwise it is decreased. Salomon in 1989 used a simple evolution strategy to adjust the global learning rate [11].

Silva and Almeida [12] proposed a new strategy to adjust local learning rates for every connection in order to maximize the update step size. This strategy increases or decreases the local learning rate by multiplying it with some constant values. A total backtracking strategy was also used to restart an update step when the error increased by using the halving of learning rates. The main shortcoming is the selection of the constant values for different problems.

Up to the present, a lot of methods are proposed by using different optimization techniques. The most well known algorithms of this type are the conjugate gradient training algorithm [13] and Levenberg-Marquardt (LM) training algorithm [14]. The computational complexity of the conjugate gradient algorithm is heavily dependent on the line search methods. The LM algorithm has a faster speed than gradient and hardly stuck in the local minimum. But it requires too much memory and computational time.

Stochastic search methods mainly include: simulated annealing (SA) [15], Alopex algorithm [18][19][20] and genetic algorithm (GA)[16][17]. SA has not been popular among neural network researchers working on feedforward neural network, except for an interesting paper by Engle [15] in which the network adaptive parameters were discretized. Alopex algorithm was introduced to the training of feedforward neural networks in 1991 [18][19]. As yet, Alopex algorithm has been tested so far only on a few problems with good results. Genetic algorithms received great popularity in neural network optimization. The selection of encoding, operators and fitness functions depends largely on experience and real problems.

3 Local Search

Local search, or local optimization, is a primitive form of continuous optimization in the discrete search space. It was one of the early techniques proposed to cope with the overwhelming computation intractability of NP-hard combinatorial optimization problems.

Given a minimization (maximization) problem with object function f and feasible region R, a typical local search algorithm requires that, with each solution point $x_i \subset R$, a there is associated a predefined neighborhood $N(x_i) \subset R$. Given a current solution point $x_i \subset R$, the set $N(x_i)$ is searched for a point x_{i+1} with $f(x_{i+1}) < f(x_i)(f(x_{i+1}) > f(x_i))$. If such a point exits, it becomes the new current solution point, and the process is iterated. Otherwise, x_i is retained as local optimum with respect to $N(x_i)$. Then, a set of feasible solution points is generated, and each of them is "locally" improved within its neighborhood. To apply local search to a particular problem, one need only to specify the neighborhood and the randomized procedure for obtaining a feasible starting solution.

Local search is very efficient in several aspects. First, at the beginning of the search, using a full assignment to all variables in the search space, it makes the search space more manageable. Secondly, it searches for improvement and, if there is any improvement, takes an action for improvement. Since the object function has a polynomial number of input numbers, both "testing" and "action" are relatively efficient and often perform well for a search problem. A major weakness of local search is that the algorithm has a tendency to get stuck at a locally optimum configuration (i.e., a local minima).

4 Proposed Algorithm

In this paper, we only deal with three-layer feedforward neural network. We take squared error as the object function. Then we can obtain the object function as follows:

$$E = \sum_p \sum_i (O_{p_i} - y_{p_i})^2 = \sum_p \sum_i (g(\sum_j w_{ij} g(\sum_k w_{jk} b_{pk})) - y_{p_i})^2 \cdot \tag{1}$$

Where p, i, j, k, g, w_{ij}, b_{pk}, y_{pi} and O_{pi} denote the pattern number, the output unit number, the hidden unit number, the input unit number, the activation function, the connection weight, the input, the target output and the real output respectively.

The proposed algorithm (see Fig.1) consists of an initialization stage and a search stage. Several operations, i.e., *GetData*(), *GetInitWeight*(), *GetError*(), *WeightPerturb*() and *OutPutResults*(), are needed in the algorithm. We will see their operations in the following discussion of the algorithm.

4.1 Initialization

Initially, the function *GetData*() gets training examples set and all parameters required in training, i.e., max iterations, max search times in each iteration, squared error for termination and number of hidden unit. The weights are chosen randomly in the range –5.0 to +5.0 by the function *GetInitWeight*(). Then the function *GetError*() calculates the squared error for the first time.

4.2 Search

The search process is an iterative local optimization process. During the *k*th iteration, for a randomly selected weight (variable), we search through its neighborhood to select a solution point that minimizes the object function. That is, during a single search in an iteration, only one weight is considered for its contribution to total squared error. The test is to see whether the assignment of a new value to the selected weight will reduce the total squared error. If the assignment would reduce the total squared error, we accepted the point as a new solution point. If not, we search again.

This process continues until a new solution is found or the search times equal the max times we have set for search in every iteration.

4.3 Termination

In practice, before error reduces to the specified error threshold for termination, the algorithm might be stuck at a locally optimum point. To settle this problem, a simple local handler that may assign new values to up to *m* weights is added to improve its convergence performance. The basic idea is to assign new values to some weights in current solution point, when the algorithm was stuck on a local minimum, accepting a modified point as a new current solution not only if the value of the cost function is better but also if it is worse.

It is quite hard and time wasting to decide whether our algorithm is absolutely trapped in a local minimum. But we can easily decide if it is possible to fall in a local minimum. In our algorithm, we adopt two criteria to judge whether it may be stuck in a local minimum. First, during consecutive constant (such as 10) iterations, the total squared error was not changed; second, during consecutive larger constant (such as 100) iterations, the relative change of the total squared error is very small. If it satisfies one of above rules, we could think it might trap in local minimum and a local handler will be executed to help it to jump out the local minimum.

4.4 Method for Reducing Running Time

In one iteration, only one selected weight (variable) value changes in each search. For a three-layer neural network, i.e., only one weight in hidden layer or one weight in output-layer is assigned a new value in each search. If the selected weigh is in hidden layer, the output of one hidden node and the outputs of all nodes in output-layer are changed; if in output-layer, only the output of one output-layer nodes is changed. Thereby, it is not necessary to calculate outputs for all nodes in the network for one test that checks whether the change of the selected weight can reduce the total error. And what we need to do is to calculate the output of the node whose output is changed when a new value is assigned to the selected weight.

```
Public MaxTrainNum As Long          'Max iterations for termination
Public RealTrainNum As Long         'current iterations
Public TrainError As Double         'squared error for termination
Public RealError As Double          'current squared error
Public MaxSearchNum As Long         'Max times for search in each iteration
Private Sub LocalSearchTrain()
  GetData()                         'get all data for training
  GetInitWeight()                   'initialize the weights
  RealError = GetError()            'calculate the squared error
  RealTrainNum = 0
  While RealError > TrainError And RealTrainNum < MaxTrainNum
      Select a weight for search randomly
      For SearchNum =1 to MaxSearchNum
         Assign a new value to the selected weight
         if the squared error is reduced Then
             Exit the cycle
         End If
      Next SearchNum
      RealError= GetError()           ' calculate the squared error
      If local Then
         WeightPerturb()              'select some weights for perturbation
      End If
      RealTrainNum = RealTrainNum + 1
  Wend
    OutPutResults()                   'output the results of training
End Sub
```

Fig. 1. The proposed algorithm

4.5 An Improved Algorithm

In each search, a new solution point is produced by assigning a new random value to the selected weight. The randomness of the assigned value makes our algorithm very powerful in searching a better solution point in a quite large range. But some good solutions may be missed because of the randomness of the assigned value.

In each search, our algorithm searches a better solution point in the neighborhood of the selected weight, i.e., the search of our algorithm is equivalent to find a minimum for a single variable function. Obviously, the single variable function is continuous. The current value of the selected weight may be minimum, maximum or a point between the minimum and the maximum of the single variable function. Apparently, the probability of being points between minimum and the maximum is the biggest. First, we choose a small positive parameter **δ,** such as 0.001, 0.01 or 0.1. A new value for the selected weight is generated by subtracting or adding the **δ**. If the current value of the selected weight is the minimum, then the new value generated by subtracting or adding will both increase the total error in quite large possibility. If the current value is the maximum, then the new value generated by subtracting or adding will both decrease the total error in quite large possibility. If the current value is between the minimum and the maximum, then the new value by either subtracting or adding the **δ** will decrease the total error in quite large possibility. If the **δ** is appropriately small, then only two searches can reduce the total error in quite large possibility. Consequently, this search method adds some deterministic factors to our algorithm.

In our improved algorithm (see Fig.2), in each iteration, the first and the second search employ above method to produce new value for the selected weight. The experimental results of the XOR and 3-Parity problem verify the efficiency of this improved algorithm.

```
Public Function SelectWeight2() As Double
  'OldWeightVal    'the old value of the selected weight in each iteration
  'Delta           'the selected small positive parameter
  If Is it the first search in each iteration Then
     SelectWeight2= OldWeightVal - Delta
     Exit Function
  ElseIf Is it the second search in each iteration Then
     SelectWeight2= OldWeightVal + Delta
     Exit Function
  End If
  'Get a new value randomly
  SelectWeight2= GetNewValRand()
End Function
```

Fig. 2. The improved producing operator

5 Experiment Results

The proposed algorithm has been tested on many classical neural network traini-ng problems such as the XOR problem, the n-Parity problem, function approximation problem, TwoNorm problem, building problem, banking problem and MushRoom problem. In this section, we give four examples: XOR problem, 3-Parity problem, TwoNorm problem and Mush Room problem. In order to investigate the performance of proposed algorithms, experiment results of the BP algorithm and BP algorithm with momentum are given in parallel. We select a three-layer neural network. In these experiments, we select the best results for BP algorithm and BP algorithm with momentum. On the contrary, we select average results for proposed algorithms since our algorithms achieve very similar results for several executions.

5.1 XOR Problem

In this problem, the network consists of two input nodes, two hidden units and one output node. Our improved algorithm was employed to train this problem and outperforms the BP algorithm and BP algorithm with momentum. After 10000 iterations, the BP algorithm only reduces the squared error to 0.5097. With a faster speed than BP algorithm, the BP algorithm with momentum reduces the squared error to 7.8469E-04. Our improved algorithm reduces the error to 4.1651E-32 at 1000th iteration and 2.7919E-33 at 10000th iteration. Therefore our improved algorithm converges faster than BP algorithm and BP algorithm with momentum.

Table 1. The performance of different algorithms on XOR problem

Algorithm / Epoch	BP	BP with Momentum	Proposed (Improved)
1000	1.0000	0.9025	4.1651E-32
2500	0.9035	6.0885E-03	1.6295E-32
5000	0.5305	1.9084E-03	1.5415E-32
10000	0.5097	7.8469E-04	2.7919E-33

5.2 3-Parity Problem

3-Parity problem is an extension of XOR problem. It comprises of 8 patterns and each pattern comprises of three inputs and one output. We select a network with four hidden units. Our improved algorithm was employed to train this problem, which outperforms the BP algorithm and BP algorithm with momentum. After 10000 iterations, the BP algorithm only reduces the squared error to 0.0181. With a faster speed than BP algorithm, the BP algorithm with momentum reduces the squared error to 4.4770E-04. Our improved algorithm reduces the error to 2.2684E-18 at 1000th iteration and 3.8991E-31 at 5000th iteration. Therefore our improved algorithm converges faster than BP algorithm and BP algorithm with momentum.

Table 2. The performaces of different algorithms on 3-Parity problem

Algorithm / Epoch	BP	BP with Momentum	Proposed (Improved)
2500	1.4986	2.4318E-03	2.2684E-18
5000	1.0161	1.0120E-03	3.8991E-31
10000	0.0181	4.4770E-04	3.8991E-31

5.3 TwoNorm Problem

TwoNorm Problem is Leo Breiman's TwoNorm example, which is Classification of 2 overlapping normal distributions [21]. There are 5300 patterns in total that are divided into the 300-member training set and the 5000-member test set. Each pattern has 20 inputs and one output. The network architecture used to learn this problem is a 20-10-1 one. From fig.3, we can see our proposed algorithm is faster than BP algorithm and BP algorithm with momentum. Further more, Table III indicates that our algorithm gives best performance (95.52%) for test example and (100%) for training examples.

Table 3. The performance of different algorithms on TowNorm problem

Algorithms	Squared Error (10000 Epoch)	Training set % right	Test set % right
BP	1.10680	99.7%	95.32%
BP with Momentum	4.1562E-02	100%	95.36%
Proposed	8.7995E-09	100%	95.52%

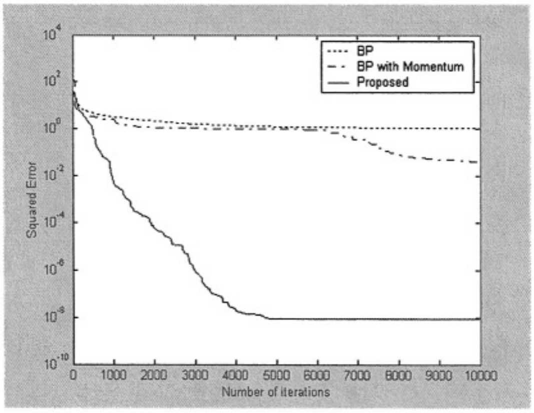


Fig. 3. The learning curves of different algorithms on TwoNorm problem

5.4 MushRoom Problem

MushRoom Problem is selected from UCI repository [22], which is a mushroom edible and poisonous classification problem. The mushroom problem is very large. There are 8124 patterns in total. We select 4062 patterns for training and the rest 4062 patterns for test. The network architecture used to train this problem is a 125-4-2 one. Fig.4 depicts learning profiles produced for this problem and indicates that the proposed algorithm yields much faster learning speed. Table IV indicates that our algorithm gives best performance (100%) for test examples and (100%) for training examples.

Table 4. The performance of different algorithms on MushRoom problem

Algorithms	Squared Error (10000 Epoch)	Training set % right	Test set % right
BP	95.5009	97.1%	96.9%
BP with Momentum	97.9782	96.6%	96.7%
Proposed	2.8501E-05	100%	100%

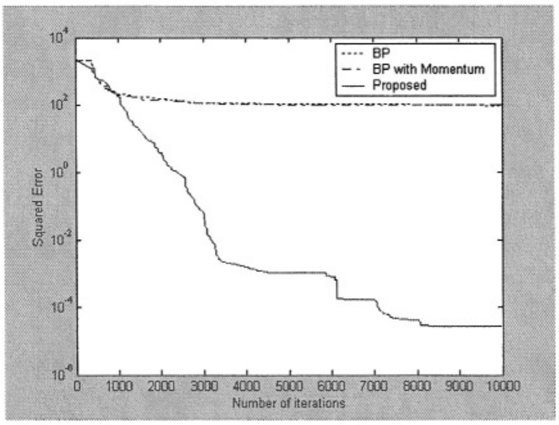


Fig. 4. The learning curves of different algorithms on MushRoom problem

6 Conclusion

An efficient learning algorithm and its improved algorithm based on local search are proposed in this paper. A number of experiments on examples ranging from small size one to large size one have been conducted. The experiments show that the proposed algorithms are able to train the feedforward neural network much faster than BP algorithm and acquire higher accuracy than BP algorithm. Our proposed algorithms provide high accuracy rates on both the training data and the testing data, which is comparable to BP algorithm.

References

1. W. S. McClulloch and W. Pitts: A logical calculus of the ideas immanent in neurons activity. Bulletin of mathematical biophysics (1943), vol.5, pp.115-133
2. J. S Judd: Neural Network Design and Complexity of Learning. MIT Press (1990)
3. Blum: Training A 3-Node Neural Network Is NP-Complete (1992)
4. Rok Sosic and Jun Gu: A polynomial time algorithm for the n-queens problem. SIGART Bulletin (1990), vol.1, pp.7-11
5. Rok Sosic and Jun Gu: Fast search algorithms for n-queens problem. IEEE Trans. System, Man and Cybernetics (1991), vol.6, pp.1572-1576
6. Rok Sosic and Jun Gu: Efficient Local Search With Conflict Minimization: A Case Study of the n-Queens Problem. IEEE Trans. System, Man and Cybernetics (1994), vol. 6, pp. 661-668
7. Jun Gu and Xiaofei Huang: Efficient Local Search With Search Space Smoothing: A Case Study of Traveling Salesman Problem (TSP). IEEE Trans. System, Man and Cybernetics (1994), vol.24, pp.728-735
8. Jun Gu: Local Search for Satisfiability (SAT) Problem. IEEE Trans. System, Man and Cybernetics (1993), vol. 23, pp.1108-1129
9. Bart Selman, Hector Levesque and David Mitchell: A New Method for Solving Hard Satisfiability Problems. Proceedings of the Tenth National Conference on Artificial Intelligence (1992), pp. 440-446,.
10. L. W., Chan and F, Fallside: An adaptive training algorithm for backpropagation networks. Computer Speech and Language (1987), vol. 2, pp.205-218
11. Salomon: Ralf Adaptive Regelung der Lernrate bei back-propagation. Forschungsberichte des Fachbereichs Informatik. Technische Universitat Berlin (1989), Bericht
12. Fernando M., Silva and B. Luis: Almeida. Speeding up Backpropagation. Advanced Neural Computers, Eckmiller R. (Editor) (1990), pp.151-158
13. Martin F. Moller: A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning. Neural Networks (1993), vol.6, pp.525-633
14. M. T. Hagan and M. B. Menhaj: Training feedforward networks with the Marquardt algorithm. IEEE Trans. Neural Network (1994), vol.5, pp.989-993
15. J. Engel: Teaching feed-forward neural networks by simulated annealing. Complex Systems (1988), vol.2, pp.641-648
16. D.L. Prados: New learning algorithm for training multilayer neural networks that uses genetic-algorithm techniques. Electronics Letters (1992), vol.28, pp.1560-1561
17. D.J. Montana: Neural network weight selection using genetic algorithms. Intelligent Hybrid Systems (1995), Wiley, New York, pp.85-104
18. A. S. Pandya and R. Sazbo: A Fast Learning Algorithm for Neural Network Applications. IEEE (1991), pp. 1569-1573
19. K. P. Venugopal and A. S. Pandya: Alopex Algorithm For Training Multilayer Neural Networks. IEEE, pp.196-201

20. K.P. Unnikrishnan and K.P. Venugopal: Alopex: a correlation-based learning algorithm for feedforward and recurrent neural networks. Neural Computations (1994), vol.6, pp.469-490
21. L. Breiman: Bias, variance and arcing classifiers. Tec. Report 460, Statistics department. University of California (1996)
22. Merz, C.J., Murphy, P.: UCI repository of machine learning database. http://www.ics.uci.edu/~mlearn/MLRepository.html.

Combining Data Reduction and Parameter Selection for Improving RBF-DDA Performance

Adriano L. I. Oliveira[1,2], Bruno J. M. Melo[1], Fernando Buarque L. Neto[1,2], and Silvio R. L. Meira[2]

[1] Polytechnic School, University of Pernambuco,
Rua Benfica, 455, Madalena, Recife – PE, Brazil, 50.750-410
[2] Center of Informatics, Federal University of Pernambuco,
P.O. Box 7851, Cidade Universitaria, Recife – PE, Brazil, 50.732-970
{alio, fbln, srlm}@cin.ufpe.br, bjmm@upe.poli.br

Abstract. The Dynamic Decay Adjustment (DDA) algorithm is a fast constructive algorithm for training RBF neural networks for classification tasks. In a previous work, we have proposed a method for improving RBF-DDA generalization performance by adequately selecting the value of one of its training parameters (θ^-). Unfortunately, this method generates much larger networks than RBF-DDA with default parameters. This paper proposes a method for improving RBF-DDA generalization performance by combining a data reduction technique with the parameter selection technique. The proposed method has been evaluated on four classification tasks from the UCI repository, including three optical character recognition datasets. The results obtained show that the proposed method considerably improves performance of RBF-DDA without producing larger networks. The results are compared to MLP and k-NN results obtained in previous works. It is shown that the method proposed in this paper outperforms MLPs and obtains results comparable to k-NN on these tasks.

1 Introduction

The Dynamic Decay Adjustment (DDA) algorithm has been originally proposed for constructive training of radial basis functions neural networks (RBFNs) for classification tasks [2]. Later, the algorithm was adapted for constructive training of probabilistic neural networks [1]. The number of hidden RBFs, the center and width of the RBFs and the weights of the connections from hidden to output layer are determined by DDA during training. The algorithm has a number of interesting characteristics. For example, it does not depend on weights initialization, training is very fast and it has achieved classification performance comparable to multi layer perceptrons (MLPs) in a number of tasks [2].

In previous works, RBF-DDA generalization performance was found to have weak dependence on training parameters θ^+ and θ^- for some datasets [2,1]. Therefore, the use of the default parameters values ($\theta^+ = 0.4$ and $\theta^- = 0.1$) is recommended [2,1]. However, we have observed experimentally that for some datasets the value of θ^- considerably influences performance [8]. Training with DDA with θ^- smaller than the default results in networks with much more hidden units than those of networks built with default θ^-. This can improve coverage of the input space and thus enhance generalization performance. Nonetheless, the value of θ^- should be carefully selected, since θ^- values too small can lead to overfitting [8].

In another previous work, a method for improving RBF-DDA performance based on the selection of an appropriate value for θ^- was proposed [8]. It has been shown that this method considerably improves RBF-DDA performance on some optical character recognition datasets. Yet, unfortunately it also leads to RBF-DDA networks with much larger number of hidden units than those of RBF-DDA networks trained with default parameters [8].

This paper proposes a method which aim to improve RBF-DDA performance without increasing resulting networks. The method proposed here employs a data reduction algorithm recently put forward for expediting support vector machines model selection [9]. In the original application, the data reduction algorithm was used to reduce the number of patterns in the training set in order to speed the selection of the training parameters of SVM, namely, the cost parameter C and the kernel parameter γ, for optimal SVM performance. These parameters are selected by cross-validation using a reduced training set instead of the full training set. This procedure results in faster model selection. Subsequently, the SVM is trained with the full training set using the optimal values of the parameters C and γ. The data reduction algorithm has a parameter k which has direct influence on the reduction rate [9].

In this paper, we combine the data reduction algorithm mentioned above with the method based on selection of the parameter θ^- in order to improve RBF-DDA performance. Firstly, the training set is reduced by using the data reduction algorithm with a suitable value for the parameter k. Next, an RBF network is built by using the DDA algorithm. Notice that, in contrast to its previous use, the data reduction algorithm is used here to produce a reduced training set that is used to train an RBF network, instead of just selecting its optimal parameters. The motivation for using the data reduction algorithm is to eliminate redundant training patterns in order to help DDA produce much smaller networks. Moreover, the experiments show that such networks generalize considerably better. The integration of DDA training with the data reduction algorithm requires selection of three parameters: θ^+, and θ^-, and k. The first one has weak influence on performance. The second and third ones are selected by cross-validation using training data. θ^- is selected as in our previous work [8].

The method proposed in this paper has been tested on four datasets obtained from the UCI machine learning repository [3]. The results show that the classification performance of RBF-DDA on test sets is much improved by the proposed method. In addition, it is shown that the proposed method outperforms previous results obtained by MLPs on all datasets considered [5,6,4] and achieves similar performance to k-nearest neighbor (k-NN) on the datasets considered

[5,6,4]. Moreover, the proposed method produces much smaller RBF-DDA networks than previous method based solely on θ^- selection [8], which can be an important advantage for practical applications.

This paper is organized as follows. In next section, we briefly review the DDA training algorithm for RBF networks. We also present the data reduction algorithm used in this work. Next, we present the proposed method for training RBF-DDA networks by combining the data reduction algorithm and the suitable selection of θ^-. Experimental results using four datasets from the UCI repository are provided in section 3. This section compares performance of the proposed method with default RBF-DDA as well as with MLP and k-NN results reported in the literature. Finally, section 4 concludes the paper.

2 The Proposed Method

2.1 Training RBF Networks with the DDA Algorithm

The DDA is a very fast constructive training algorithm for RBF networks [2]. RBF-DDAs have a single hidden layer, whose number of units is automatically determined during training. The training algorithm is constructive, starting with an empty hidden layer, with units being added to it as needed. RBF-DDA uses Gaussians as activation functions for the units in the hidden layer.

The activation $R_i(\vec{x})$ of a hidden unit i is given by $R_i(\vec{x}) = \exp\left(-\frac{||\vec{x}-\vec{r_i}||^2}{\sigma_i^2}\right)$, where $\vec{x}$ is the input vector and $||\vec{x} - \vec{r_i}||$ is the Euclidian distance between the input vector $\vec{x}$ and the Gaussian center $\vec{r_i}$. $\vec{r_i}$ and σ_i values are determined automatically during training (see algorithm 1) [2].

The number of units in the input layer represents the dimensionality of the input space. The input layer is fully connected to the hidden layer. RBF-DDA uses 1-of-n coding in the output layer, where each unit of this layer represents a class. Classification uses a winner-takes-all approach, where the unit with the highest activation gives the class. Each hidden unit is connected to exactly one output unit. Each connection has a weight A_i whose values are also revealed by the training algorithm. Output units uses linear activation functions with values computed by $f(\vec{x}) = \sum_{i=1}^{m} A_i \times R_i(\vec{x})$, where m is the number of RBFs connected to that output.

The DDA algorithm relies on two parameters in order to decide about the introduction of new prototypes (RBF units) in the networks. One of these parameters is a *positive threshold* (θ^+), which must be overtaken by an activation of a prototype of the same class so that no new prototype is added. The other is a *negative threshold* (θ^-), which is the upper limit for the activation of conflicting classes [2].

The DDA training method for one training epoch is presented in algorithm 1 [2]. In most problems training is finished in four to five epochs [2]. Notice that during training a new RBF unit is added to the network only if there is no RBF unit from the same class of the current training pattern which gives an output greater than θ^+. In this case the DDA will set the weight of the

Algorithm 1 DDA algorithm for RBF training

```
1: // reset weights:
2: FORALL prototypes $p_i^k$ DO
3:     $A_i^k = 0.0$
4: ENDFOR
5: // train one complete epoch
6: FORALL training pattern $(\vec{x}, c)$ DO
7:     IF $\exists p_i^c : R_i^c(\vec{x}) \geq \theta^+$ THEN
8:         $A_i^c + = 1.0$
9:     ELSE
10:        // "commit": introduce new prototype
11:        add new prototype $p_{m_c+1}^c$ with:
12:        $\vec{r}_{m_c+1}^c = \vec{x}$
13:        $\sigma_{m_c+1}^c = \max_{k \neq c \wedge 1 \leq j \leq m_k} \{\sigma : R_{m_c+1}^c(\vec{r}_j^k) < \theta^-\}$
14:        $A_{m_c+1}^c = 1.0$
15:        $m_c + = 1$
16:    ENDIF
17:    // "shrink": adjust conflicting prototypes
18:    FORALL $k \neq c, 1 \leq j \leq m_k$ DO
19:        $\sigma_j^k = \max\{\sigma : R_j^k(\vec{x}) < \theta^-\}$
20:    ENDFOR
21: ENDFOR
```

new RBF unit to 1.0. Otherwise, an existing RBF unit of the same class of the current training pattern and with output greater than θ^+ will be associated to the current training pattern and its weight will be incremented (line 8 of algorithm 1). Hence, all weights of a trained RBF-DDA networks have integer values. A trained RBF-DDA network holds the following two equations for every training pattern $\vec{x}$ of class c, $\exists i : R_i^c(\vec{x}) \geq \theta^+$ and $\forall k \neq c, 1 \leq j \leq m_k : R_j^k < \theta^-$.

2.2 Data Reduction Algorithm

The data reduction algorithm employed in this work was originally proposed for speeding up model selection for support vector machines [9]. In that context, the algorithm is used to reduce the number of patterns of the training set. Subsequently, the reduced training set is used to select the values of SVM parameters, C – the cost parameter, and γ – the kernel parameter, which optimize performance [9]. Finally, these selected parameters are used to train a SVM using the complete training set (without reduction).

The data reduction algorithm is presented below (see algorithm 2). The idea of the algorithm is to reduce the training set by removing those training patterns that are far away from the boundary that separate different classes of training data. The algorithm works by first sorting the training dataset in descending order according to the distance of each pattern's *nearest enemy* [9]. The nearest enemy of a pattern is defined as the nearest neighbor of the pattern that belongs to a different class. Next, the patterns are examined for removal beginning at the pattern furthest from its nearest enemy. Finally, the data reduction algorithm examines patterns in the ordered training set one by one, removing a pattern if the pattern and all of its k nearest neighbors in the remaining training set belong to the same class. In general, larger values of k result in lower reduction rates (fewer training patterns are removed from the training set). In the context of SVM model selection, $k = 3$ was used [9].

Algorithm 2 The data reduction algorithm

```
Input: Training dataset T
Output: Reduced dataset R
// $NED(x_i)$ : the distance of $x_i$'s nearest enemy
// $N_k(x_i)$ : the set of $x_i$'s nearest neighbors
// $C(x_i)$ : the class label of $x_i$

Data Reduction(T)
1:   $R \leftarrow T$
2:   FOREACH instance $x_i$ in $R$
3:     Find $NED(x_i)$
4:   SORT $R$ in descending order by $NED(x)$
5:   FOREACH instance $x_i$ in $R$
6:     Find $N_k(x_i)$ in $R$
7:     IF $C(x_i) == C(p), \forall p \in N_k(x_i)$
8:       Remove $x_i$ from $R$
9:   RETURN $R$
```

2.3 Training RBF-DDA by Combining Data Reduction and Parameter Selection

The method proposed in this paper works by first applying the data reduction algorithm to the training set. Next, an RBF-DDA is trained by using the reduced training set. The performance of the RBF-DDA built by the method depends on k (the parameter of the data reduction algorithm) and θ^+ and θ^- (the DDA parameters).

In previous works it was shown that θ^+ and θ^- had weak influence on generalization performance for some datasets [2,1]. However we have observed that θ^- considerably influences performance for others datasets [8]. The value of θ^- influences the number of RBF units added by DDA during training. Training with smaller θ^- can produce large number of RBF units. This can lead to a classifier that correctly classifies all training samples, but that does not generalizes well. To avoid overfitting training data, the value of θ^- should be carefully selected by cross-validation. In our previous work, we have shown that one should start with $\theta^- = 0.1$ and then decrease it by $\theta^- = \theta^- \times 10^{-1}$. This should be done until the cross-validation error starts to increase [8]. We have shown previously that this simple procedure can be used to selected the near optimal θ^- for a given dataset [8]. In the previous experiments we have observed that performance does not change significantly for intermediate values of θ^-. For example: there is a significant difference in performance when θ^- decreases from $\theta^- = 10^{-3}$ to $\theta^- = 10^{-4}$, however, the search of θ^- between those values is not needed since performance does not change much.

The experiments presented below show that the parameter k of the data reduction algorithm has great influence on performance. Smaller value of k results in large reduction rates. This can result in the removal of important information for DDA training. On the other hand, larger values of k produce larger training sets, that is, smaller reduction rates. We have observed empirically that the optimal value for k in our method is near the k which reduces the training set by half. Thus, the search starts by using this value of k (which is different for each dataset) and then other values in its vicinity are tried. For a given k, θ^- is searched as explained above.

3 Experiments

The training method proposed in this paper was tested on four classification problems whose datasets are available from the UCI machine learning repository [3]. The two first datasets *(optdigits* and *pendigits)* are optical character databases of handwritten digits (one is optical and the other is pen-based) [5]. The test sets in these datasets are formed by data obtained from independent writers, that is, writers whose data are not present on the respective training sets. The third dataset *(letter)* is a database of letters that contains 20,000 patterns from 26 classes, corresponding to the 26 characters (letters) [4,5,6]. Finally, the fourth dataset *(satimage)* was generated from Landsat Multi-Spectral Scanner image data. Each pattern from this dataset contains 36 pixel values and a number indicating one of the six class categories of the central pixel [7,10]. Table 1 summarizes the main characteristics of each dataset.

Table 1. Characteristics of the datasets

	No of	No of	No of Patterns	
Dataset	Inputs	Classes	Training set	Test set
optdigits	64	10	3,823	1,797
pendigits	16	10	7,494	3,498
letter	16	26	15,000	5,000
satimage	36	6	4,435	2,000

3.1 Results and Discussion

Tables 2, 3, 4, and 5 compares performance of the method proposed in this paper with RBF-DDA trained without data reduction for *optdigits, pendigits, letter,* and *satimage,* respectively. Each of these tables contains k (the parameter of the data reduction algorithm), θ^+, θ^-, the number of hidden units of the RBF

Table 2. Classification errors on test set of *optdigits* for RBF-DDA classifiers

k	θ^+	θ^-	hidden RBF units	% tr. patterns stored	Test set error
No reduction	0.4	0.1	1,953	51.08%	10.18%
No reduction	0.4	10^{-5}	3,812	99.71%	2.78%
27	0.4	10^{-2}	1,628	42.58%	1.78%

Table 3. Classification errors on test set of *pendigits* for RBF-DDA classifiers

k	θ^+	θ^-	hidden RBF units	% tr. patterns stored	Test set error
No reduction	0.4	0.1	1,427	19.04%	8.12%
No reduction	0.4	10^{-5}	5,723	76.37%	2.92%
18	0.4	10^{-2}	1,430	19.08%	4.86%
60	0.4	10^{-2}	2,338	31.20%	2.94%
60	0.4	10^{-4}	3,130	41.77%	2.63%
80	0.4	10^{-5}	3,689	49.22%	2.60%

Table 4. Classification errors on test set of *letter* for RBF-DDA classifiers

k	θ^+	θ^-	hidden RBF units	% tr. patterns stored	Test set error
No reduction	0.4	0.1	7,789	51.93%	15.60%
No reduction	0.4	10^{-4}	12,861	85.74%	5.30%
12	0.4	10^{-2}	7,826	52.17%	5.18%
16	0.4	10^{-2}	8,582	57.21%	5.12%

network built by DDA, the percentage of training patterns stored as RBF units and the classification error on the test set. Notice that the percentage of training patterns stored as RBF units is computed with respect to the full training set.

The first line of each table presents results obtained by RBF-DDA using default parameters without data reduction. The results of the second line were also obtained by RBF-DDA without data reduction, this time trained with the previous θ^- selection method [8]. The remaining lines present the results obtained by firstly reducing the training set and then training with DDA. The selection of k and θ^- was carried out by cross-validation, as outlined in section 2. For some datasets more than one combination of k and θ^- is presented. The combination which yielded the best performance is shown on the last line of each table.

The results of tables 2, 3, 4, and 5 show that the previous method of θ^- selection considerably improves RBF-DDA performance for the four datasets considered, however, this method always leads to larger networks than those generated by RBF-DDA trained with default parameters [8]. These tables also show that the method proposed in this paper achieves better generalization performance than both default RBF-DDA and RBF-DDA trained with selected θ^- [8]. Notice that, at the same time, the proposed method is able to produce

Table 5. Classification errors on test set of *satimage* for RBF-DDA classifiers

k	θ^+	θ^-	hidden RBF units	% tr. patterns stored	Test set error
No reduction	0.4	0.1	2,812	63.40%	14.95%
No reduction	0.4	10^{-4}	4,099	92.42%	8.55%
46	0.4	10^{-3}	2,759	62.21%	8.55%
75	0.4	10^{-3}	3,090	69.67%	8.25%

much smaller networks than those produced by the RBF-DDA trained with θ^- selection (without data reduction).

For example, for the dataset *optdigits* (results in table 2), RBF-DDA trained with the full training set and default parameters achieve 10.18% classification error on test set with 51.08% of the training patterns stored as RBF units. The method based only on θ^- selection (without data reduction) is able to improve the generalization performance, reducing the classification error to 2.78%. This is obtained, however, at the expense of increasing the percentage of stored training patterns to 99.71%. The method proposed in this paper, in contrast, improves performance (the error is decreased to 1.78%) and, at the same time, decreases the network size (the number of training patterns stored is 42.58% in this case).

Table 6 compares the generalization performance of the proposed method with others classifiers for each dataset. This table presents results obtained with RBF-DDA using data reduction; RBF-DDA without data reduction using θ^- selection [8]; RBF-DDA without data reduction using default parameters; multi-layer perceptrons (MLP) and k-nearest neighbor (k-NN). As commented before, the method proposed in this paper outperforms RBF-DDA trained with both default parameters and selected θ^-. Furthermore, the proposed method produces much smaller networks than those of RBF-DDA trained with selected θ^- and comparable in size to those of default RBF-DDA.

The MLP and k-NN results for the OCR datasets shown in table 6 were obtained by a previous work [5]. In that work, MLPs with 10 hidden units and k-NN with $k = 3$ were used [5]. A number of classifiers have been compared in that work, including machine committees. k-NN was the best classifier on the *pendigits* and *optdigits* datasets [5]. A cascaded classifier formed by MLP and

Table 6. Classification errors on test sets: comparison of different classifiers

Dataset	RBF-DDA (proposed method)	RBF-DDA (θ^- *opt.*)	RBF-DDA Default	MLP	k-NN
optdigits	**1.78%**	2.78%	10.18%	10.95%	3.51%
pendigits	2.60%	2.92%	8.12%	4.83%	**2.29%**
letter	**5.12%**	5.30%	15.60%	23.59%	6.62%
satimage	**8.25%**	8.55%	14.95%	13.90%	9.40%
mean across datasets	4.44%	4.89%	12.21%	13.32%	5.45%

k-NN performed slightly better than single k-NN on the *letter* dataset [5]. The error obtained by the cascaded classifier was 6.27% which is still worse than the result obtained by the RBF-DDA classifier with data reduction and θ^- selection proposed here (5.12%).

The *letter* dataset was used previously in other works as well. The original work using this dataset has obtained around 20% classification error on the test set [4]. A more recent work has obtained 10.10% classification error on the test set using k-NN and 20.70% using an MLP as the classifier [6]. The best result obtained by the Bayesian pairwise classifiers proposed in that work was 12.40% of classification error on the test set [6]. On the other hand, the method proposed in this paper has achieved 5.12% classification error (as shown in table 6) and therefore, outperforms also all classifiers considered in that previous work [6]. Our method also outperforms the *Locally Confident Network* (LCN), which achieved 7.40% classification error on the test set [10].

The MLP and k-NN results for the *satimage* dataset were obtained also from previous works [7,10]. Our method outperforms both these classifiers in this dataset as shown in table 6. It also outperforms the LCN classifier, which achieved 9.40% classification error on test set [10].

The training method for RBF networks proposed in this work markedly outperforms MLPs in all datasets considered, as can be seen in table 6. The proposed method outperformed k-NN on three of the datasets and had similar performance on the remaining *(pendigits)*. The proposed method has an important advantage over k-NN. k-NN needs to store all training data whereas the proposed method needs to store only from 42.58% to 69.67%, depending on the dataset (see tables 2, 3, 4, and 5).

4 Conclusion

This paper has proposed a method for improving the generalization performance of RBF-DDA by combining a data reduction algorithm with a method for selecting the value of one of the DDA training parameters, θ^-.

The method proposed in this work firstly applies a data reduction algorithm to the training set. The data reduction algorithm employed has been originally proposed for expediting model selection for support vector machines. The algorithm has a parameter k which has direct influence on the reduction rate. Next, an RBF-DDA is trained by using the reduced training set. The optimal value of θ^- is selected by using a method previously proposed [8]. The optimal k is selected by cross-validation. The experiments have shown that the optimal k is near the k that reduces the training set to half.

The proposed method has been evaluated on four classification tasks whose datasets were obtained from the UCI repository [3]. The results show that the proposed method considerably improves RBF-DDA performance; the mean classification error across the datasets was 4.44%. In contrast, RBF-DDA without data reduction and trained with default parameters obtained 12.21%. The proposed method has also outperformed a previous method for improving RBF-

DDA performance based solely on θ^- selection [8]. The latter method has obtained a mean classification error of 4.89%. In addition, the experiments have shown that this past method produces networks that are much larger than those produced by the method presented in this work.

The proposed method has also outperformed MLP on all datasets (mean classification error 13.32%) and has achieved similar performance to k-NN (mean classification error 5.45%), with the advantage of producing much smaller classifiers, which can be a very important advantage for practical applications.

References

1. M. Berthold and J. Diamond. Constructive training of probabilistic neural networks. *Neurocomputing,* 19:167–183, 1998.
2. Michael R. Berthold and Jay Diamond. Boosting the performance of RBF networks with dynamic decay adjustment. In G. Tesauro et al, editor, *Advances in Neural Information Processing,* volume 7. MIT Press, 1995.
3. C. Blake and C. Merz. UCI repository of machine learning databases. Available from [http://www.ics.uci.edu/~mlearn/MLRepository.html], 1998.
4. P. W. Frey and D. J. Slate. Letter recognition using holland-style adaptive classifiers. *Machine Learning,* 6(2), mar 1991.
5. C. Kaynak and E. Alpaydin. Multistage cascading of multiple classifiers: One man's noise in another man's data. In *Proc. of the 17th International Conference on Machine Learning,* 2000.
6. S. Kumar, J. Ghosh, and M. Crawford. A bayesian pairwise classifier for character recognition. In Nabeel Mursheed, editor, *Cognitive and Neural Models for Word Recognition and Document Processing.* World Scientific Press, 2000.
7. D. Michie, D. J. Spiegelhalter, and C. C. Taylor, editors. *Machine Learning, Neural and Statistical Classification.* Ellis Horwood, 1994.
8. A. L. I. Oliveira, F. B. L. Neto, and S. R. L. Meira. Improving RBF-DDA performance on optical character recognition through parameter selection. In *Proc. of the 17th International Conference on Pattern Recognition (ICPR'2004),* Cambridge, UK, 2004. IEEE Computer Society Press. Vol. 4, pages 625–628 (see

).
9. Y.-Y. Ou, C.-Y. Chen, S.-C. Hwang, and Y.-J. Oyang. Expediting model selection for support vector machines based on data reduction. In *Proc. of IEEE Conference on Systems, Man and Cybernetics,* 2003.
10. J. Wang, P. Neskovic, and L. N. Cooper. Partitioning a feature space using a locally defined confidence measure. In *Proc. of International Conference on Artificial Neural Networks (ICANN'2003),* 2003.

Bidirectional Neural Network for Clustering Problems

Enrique Domínguez and José Muñoz

Department of Computer Science, E.T.S.I. Informatica, University of Malaga, Campus de Teatinos, s/n. 29071 Malaga, Spain
{enriqued, munozp}@lcc.uma.es

Abstract. There exist several neural networks models for solving NP-hard combinatorial optimization problems. Hopfield networks and self-organizing maps are the two main neural approaches studied. Criticism of these approaches includes the tendency of the Hopfield network to produce infeasible solutions and the lack of generalization of the self-organizing approaches. This paper presents a new bidirectional neural model for solving clustering problems. Bidirectional associative memory (BAM) is the best bidirectional neural architecture known. Typically, this model has ever been used for information storage. In this paper we propose a new neural model with this bidirectional neural architecture for optimization problems, concretely clustering problems. A sample theoretical clustering problem as the *p*-median problem is used to test the performance of the proposed neural approach against genetic algorithms and traditional heuristic techniques.

1 Introduction

The idea of using neural networks to provide solutions to difficult NP-hard optimization problems has been pursued for over decades. Hopfield and Tank [14] showed that the traveling salesman problem (TSP) could be solved using a Hopfield neural network. The technique proposed by Hopfield and Tank requires minimization of an energy function containing several terms and parameters, yet it was shown to often yield infeasible solutions to the TSP. Research efforts tried to either modify the energy function or optimally tune the numerous parameters involved, so that the network would converge to a feasible solution.

Results have shown that, unless the TSP is Euclidean, the quality of the solutions found using a Hopfield network is unlikely to be comparable to those obtained using traditional techniques [11]. Other researches [25] argued that the TSP may not be an appropriate benchmark problem anyway, due to the existence of an alternative linear formulation, which makes comparisons unfair and biases the findings against neural and other techniques using a nonlinear formulation. Although the performance of neural networks for solving optimization problems has been relatively untested, for many NP-hard optimization problems, heuristics approaches are employed due to the need for rapid solutions. Certainly, one of the principal advantages of neural networks is the rapid computation power and speed due to intrinsic parallelism and simple computation. Moreover, a hardware implementation can be made easily and increase the speed and the rapid computation power.

A similar focus on the TSP is found in the literature relating to the use of self-organizing approaches to optimization [9]. In this case, the reason is because the majority of these approaches are based upon the elastic net method [8]. Kohonen's principles of self-organization are combined with the concept of elastic band containing a circular ring of neurons, which move in the Euclidean plane of the TSP cities, so that the elastic band passes eventually through of all the cities, and represent the final TSP tour. Such approach rely upon the fact that the elastic band can move in Euclidean space, and that physical distances between the neurons and the cities can be measured in the same space.

Associative memory is one of the important artificial neural networks with a wide range of applications in areas such as content-addressable memory, pattern recognition, and intelligent control. Among the many associative memory models, bidirectional associative memory (BAM) proposed by Kosko [19] has given rise to much interest because of its capability of achieving heteroassociation with a smaller correlation matrix. Such a model is potentially powerful in pattern classification and pattern identification. Many discussion and modifications are available [2]. They often give improved performance introducing new coding methods or modifying learning, though some are problem dependent or with extra costs.

Although BAM is a promising model, its information processing mechanism still has no been fully understood. It has been shown that the Hopfield autoassociative memory model is equivalent, and we will generalize this idea to the BAM using Hebbian correlation encoding method[2].

In this paper, we will show that the BAM can be used for solving clustering problems. The bidirectional associative memory is a simple kind of heteroassociative memory and can be viewed as an extension of Hopfield autoassociative memory from one-layer unidirectional network to two-layer bidirectional structure. In this sense, BAM can be used as Hopfield networks to solve optimization problems. The true advantage of using recurrent neural networks as BAM or Hopfield networks to solve difficult optimization problems relates to speed considerations. Due to their inherently parallel structure and simple computational requirements, neural network techniques are especially suitable for direct hardware implementation, using analog or digital integrated circuits [16] [20], or parallel simulation [23].

The problems associated with the recurrent neural networks include the tendency of the energy function to settle in poor local minima, often one that does not represent a feasible solution to the optimization problem, and the difficulty in selecting appropriate parameters in the energy function. Many researchers have tried to optimally select these parameters [24], while others have focused their attention on reducing the number of terms in the energy function. In this paper, a bidirectional neural network with a competitive dynamics avoiding the parameter tuning is proposed for solving clustering problems. A sample theoretical clustering problem as the p-median problem is used to test the performance of the proposed neural approach against traditional heuristic techniques.

The outline of this paper is as follows. Section 2 and 3 briefly reviews the underlying principles of the clustering and BAM, respectively. The proposed bidirectional neural network is presented in section 4. A description of the computer simulation follows, together with presentation of the results, in section 5. Finally, section 6 provides a summary and conclusions.

2 Clustering Problems

Clustering, so called set partitioning, is a basic and widely applied methodology. Application fields include statistic, computer science (such as pattern recognition, learning theory, image processing and computer graphics, etc.) and mathematical programming (including network partitioning, routing, assignment problems, scheduling and location problems, etc.). Clustering procedures use a criterion function, such as the sum of the squared distances from the cluster centers, and seek the grouping that optimizes the criterion function. That is, clustering problems consist on finding natural groupings in a set of data. We define a natural grouping as a partition that involves that the similarity between samples in the same cluster is significantly less than the similarity between samples in different clusters and optimizes the criterion function of the problem. The mathematical model of clustering problems, given the number of clusters, is defined as follows

Optimize

$$F(X) \tag{1}$$

Subject to

$$\sum_{q=1}^{p} x_{iq} = 1 \quad i = 1,2,\dots n \tag{2}$$

where

$F(X)$ is the criterion function of the problem
n is the number of samples
p is the number of cluster
$$x_{iq} = \begin{cases} 1 & \textbf{if } i \textbf{ is assigned to the cluster } q \\ 0 & \textbf{otherwise} \end{cases}$$

The most obvious measure of the similarity (or dissimilarity) between two samples is the distance between them. One way to begin a clustering investigation is to define a suitable metric and compute the matrix of distances between all pairs of samples. For instances, one broad class of metrics is the general Minkowski metric defined by expression (3). Setting $k = 2$ gives the familiar Euclidean metric while setting $k = 1$ gives the Manhattan metric.

$$d(\bar{x}, \bar{y}) = \sqrt[k]{\sum_i |x_i - y_i|^k} \tag{3}$$

The simplest and most widely used criterion function for measuring the clustering quality of any partition is the sum of squared error criterion defined by

$$F(X)=\sum_{i=1}^{n}\sum_{q=1}^{p}\left\|X_i-M_q\right\|^2 x_{iq} \tag{4}$$

where M_q is the cluster center, that is, the best representative of the samples in the cluster q that minimizes the sum of squared error. In this sense the cluster center M_q is the mean of cluster. Other criterion function can be obtained replacing M_q by the median or evaluating the error between the cluster center and the farthest pattern.

In this paper, we examine the BAM application to solve a classical clustering problem as the *p*-median problem. The *p*-median problem is a well-known location problem that has been studied during years. This problem is concerned the location of *p* facilities (medians) in order to minimize the total weighted distance between the facilities and the demand points (population centers or customers). Kariv and Hakimi [17] showed that the *p*-median problem on a general network is NP-hard. So, a lot of heuristic solution techniques have been developed in an effort to solve large-scale problems to near-optimality with a reasonable computational effort. A number of solution procedure have been developed for general networks. Most of the proposed procedures have been based on mathematical programming relaxation and branch-and-bound techniques. However, recently have been developed new procedures based on tabu search, genetic algorithms and neural networks. Thus, some proposed procedures include tree search [3] [4], lagrangian relaxation coupled with branch & bound [10][18], tabu search [21], as well as other heuristic and decision techniques [6][15][26]. Other proposed techniques more recent include heuristic concentration [22], variable neighborhood search [13] and genetic algorithms [1].

3 Bidirectional Associative Memories

A discrete BAM is a two-layer nonlinear recurrent neural network. One layer is denoted as field *A* with processing elements or neurons A_i and the other layer is denoted as field *B* with processing elements B_j. Layers *A* and *B* consist of N_A and N_B neurons or processing elements, respectively. Output neuron states A_i and B_j take on the values 0 and 1, and the state of the BAM is a binary (N_A+N_B) tuple. The connecting link between two neurons A_i and B_j has associated a synaptic weight w_{ij}, which is determined during the learning phase and fixed during the retrieval phase. This class of neural networks is characterized by information storage and recall. There are some applications in areas such as content-addressable memory, pattern recognition, and intelligent control.

In this paper, however, we use a discrete BAM as optimization machine; therefore the learning phase is not used. The neural approach is designed formulating the optimization problem in terms of minimizing a cost or energy function, which any global minimum is an optimal solution of the problem. The BAM energy function is defined as follows

$$E = \sum_{i=1}^{N_A} \sum_{j=1}^{N_B} w_{ij} A_i B_j \tag{5}$$

Unlike Hopfield network, which may have oscillatory states in synchronous mode, BAM can guarantee that a final stable state can be reached. However, a pattern can be recalled if and only if this pattern is a local minimum of the energy function [19]. In order to design a suitable neural network for clustering problems, the key step is to define an appropriate energy function E that optimizes the criterion function $F(X)$ of the clustering problem. Thus, any global minimum of the energy function will be an optimal solution of the clustering problem.

$$\text{allocation layer} \left\{ \begin{array}{cccccc} x_{11} & x_{12} & \cdots & x_{1p} & \rightarrow & \text{group 1} \\ x_{21} & x_{22} & \cdots & x_{2p} & \rightarrow & \text{group 2} \\ \vdots & \vdots & \ddots & \vdots & & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} & \rightarrow & \text{group n} \end{array} \right.$$

$$\text{location layer} \left\{ \begin{array}{cccc} y_{11} & y_{12} & \cdots & y_{1p} \\ y_{21} & y_{22} & \cdots & y_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ y_{n1} & y_{n2} & \cdots & y_{np} \\ \downarrow & \downarrow & & \downarrow \\ \text{group n}+1 & \text{group n}+2 & & \text{group n}+\text{p} \end{array} \right.$$

Fig. 1. Neuron organization of the proposed BAM

4 Bidirectional Neural Model for Clustering

The proposed BAM for clustering problems consists of two layers of np binary neurons or processing elements, where n is the number of patterns and p is the number of cluster. The first layer X is named allocation layer and is composed of np allocation neurons (x_{iq}) representing either the pattern i is assigned to cluster q or not. The second layer Y is named location layer and is composed of np location neurons (y_{jq}) indicating either the pattern j is the center of cluster q or not. The output neuron states x_{iq} (y_{jq}) depends on their activation potentials $h_{x_{iq}}$ ($h_{y_{jq}}$), which are a weighted linear combination of the output states of other neurons. Therefore, the output state of a neuron depends of the output states of all the neurons.

In order to guarantee a feasible solution of the clustering problem and avoid the parameter tuning problem, each layer of the proposed BAM is organized in disjoint competitive groups of neurons. In each group only one neuron can be activated at the same time (the winner neuron). Therefore, as shown in Fig. 1, n competitive groups form the allocation layer, and q competitive groups form the location layer.

In this paper, we concern with solving the classical p-median problem. Thus, the criterion function of the p-median problem to be optimized, that is, the energy function of the neural network, is defined as follows

$$E = \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{q=1}^{p} d_{ij} x_{iq} y_{jq} \tag{6}$$

where

n is the number of patterns

p is the number of clusters

d_{ij} is dissimilarity between patterns i and j

$$x_{iq} = \begin{cases} 1 & \textbf{if pattern } i \textbf{ is assigned to the cluster } q \\ 0 & \textbf{otherwise} \end{cases}$$

$$y_{jq} = \begin{cases} 1 & \text{if } j \textbf{ is the center of the cluster } q \\ 0 & \textbf{otherwise} \end{cases}$$

The output state of a neuron depends on its activation potential, which it is given by the following expressions

$$h_{x_{iq}}(k) = -\sum_{j=1}^{n} d_{ij} y_{jq}(k) \tag{7}$$

$$h_{y_{jq}}(k) = -\sum_{i=1}^{n} d_{ij} x_{iq}(k) \tag{8}$$

where $h_{x_{iq}}$ is the activation potential of the allocation neuron x_{iq}, and $h_{y_{jq}}$ is the activation potential of the location neuron y_{jq}. Note that $h_{x_{iq}}$ measures the dissimilarity between the pattern i and the center of cluster q, and $h_{x_{iq}}$ evaluates the dissimilarity between the cluster center j and all the patterns assigned to the cluster q.

The update of neuron states is defined according to the dynamical rules of neural network. The next state (at time $k+1$) of a neuron depends on its current activation potential (at time k) and it is defined by the following computational dynamics

$$x_{iq}(k+1) = \begin{cases} 1 & \text{if } h_{x_{iq}}(k) = \max\limits_{1 \le j \le n} \{h_{x_{jq}}(k)\} \\ 0 & \textit{otherwise} \end{cases} \tag{9}$$

$$y_{jq}(k+1) = \begin{cases} 1 & \text{if } h_{y_{jq}}(k) = \max\limits_{1 \le i \le n} \{h_{y_{iq}}(k)\} \\ 0 & \textit{otherwise} \end{cases} \tag{10}$$

Observe that only one neuron of the same group can be activated, and the neurons inside same layer are updated in parallel. The central property of the proposed network is that the computational energy function (6) always decrease (or remains constant) as the network evolve according to its computational dynamics (9) and (10).

Theorem 1. *Let M be a BAM characterized by an energy function defined by* ***(6)*** *where the activation potential of the neurons are computed by* ***(7)*** *and* ***(8)****. The energy function E decreases if the layers are updating at every time using the computational dynamics of the neural network given by expressions* ***(9)*** *and* ***(10)****.*

Proof. If we consider discrete time dynamics then the increment of energy for any change of the state of any neuron of the network is given by the next expression

$$\Delta E(k) = E(k+1) - E(k) = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{q=1}^{p} d_{ij}x_{iq}(k+1)y_{jq}(k+1) - \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{q=1}^{p} d_{ij}x_{iq}(k)y_{jq}(k)$$

We introduce now the notion of layer update, that is, instead of selecting a single neuron for update we can select a layer containing a number of *np* neurons. Then, the difference in the energy that would result if only the state of the neurons in the selected layer was altered depends on the type of layer (allocation or location layer). If all neurons contained in the selected layer are allocation neurons (x_{iq}), the difference in the energy is

$$\Delta E(k) = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{q=1}^{p} d_{ij}\left(x_{iq}(k+1) - x_{iq}(k)\right)y_{jq}(k) = \sum_{i=1}^{n}\sum_{q=1}^{p}\left(x_{iq}(k) - x_{iq}(k+1)\right)h_{x_{iq}}(k)$$
$$= \sum_{i=1}^{n}\left[\left(x_{is_i}(k) - x_{is_i}(k+1)\right)h_{x_{iq_i}}(k) + \left(x_{ir_i}(k) - x_{ir_i}(k+1)\right)h_{x_{rq_i}}(k)\right]$$

Let us suppose now that at time *k* the allocation neurons x_{ir_i} is the only one that is “on” in group *i* and that allocation neuron x_{is_i} are the candidate neurons in group *i* that is going to be “on” at time *k*+*1*. Then we have that

$$x_{ir_i}(k) = 1 \qquad x_{iq}(k) = 0 \quad 1 \le q \le p, q \ne r_i$$
$$x_{is_i}(k+1) = 1 \qquad x_{iq}(k) = 0 \quad 1 \le q \le p, q \ne s_i$$

By substituting these values we have that

$$\Delta E(k) = \sum_{i=1}^{n}\left(h_{x_{ir_i}}(k) - h_{x_{is_i}}(k)\right) \le 0$$

Hence, we have that if the neuron with the maximum input $h_{x_{is_i}}$ per group *i* is always selected as the candidate neuron x_{is_i} , then the energy descent is guaranteed.

□

5 Experimental Results

A random number generator has been used to generate two-dimensional points. All points are distributed uniformly within the unit square. The computation of the optimal solutions has been carried out with an exact algorithm proposed by Hanjoul and Peeters [12], using branch & bound.

We have compared the performance of the proposed bidirectional neural model (BNM) with the performance of the genetic algorithm (GA) proposed by Bozkaya, Zhang and Erkut [7], and with the interchange algorithm proposed by Teizt and Bart [26] (T&B) with a number of enhancements designed to accelerate the algorithm proposed by Densham and Rusthon [5]. Although T&B is a traditional heuristic, we elected this algorithm since T&B is very simple to understand and implement, and it produces good solutions with limited computational effort.

We first compare our implementation of BNM with T&B and GA on several small-scale problems. Table 1 lists the comparison of results from different algorithms with optimal solutions. For each instance, which it is represented by a row in the table, 50 randomly problems are generated and tested. The three algorithms report 100% of optimality for their 50 randomly generated problems with 50 demand points (n=50) and 5 medians (p=5). The average error figures in the table represent the average percentage deviation from the best solution calculated by an exact algorithm [12].

Table 1. Results for BNM, T&B and GA applied to small-scale problems

n	P	Avg. Error (%)		
		BNM	T&B	GA
50	5	**0.00**	**0.00**	**0.00**
50	10	**0.00**	0.03	0.02
50	20	**0.00**	0.11	0.10
75	5	**0.00**	**0.00**	0.01
75	10	**0.00**	0.09	0.10
75	20	**0.10**	0.17	0.11
100	5	**0.00**	0.05	0.02
100	10	**0.01**	0.21	0.10
100	20	**0.09**	0.39	0.23

T&B finds good solutions quickly but may be trapped in local minimum because of its myopic nature. GA used a string of p integers to represent a feasible p-median solution. In [7] (chapter 6) Bozkaya et al. found that keeping the best solution in every generation is particularly important for the *p-median* problem. When allowed to run for a sufficiently long time, GA outperforms T&B.

We tested all algorithms on an Origin 2000 computer (Silicon Graphics Inc.) with 16 CPUs MIPS R10000. In Table 2 we compare the time of the three algorithms for the same problems. In this table we show that our implementation of BNM is an intermediate alternative, since produce better results than T&B but the BNM computation time is greater than T&B. That is, the neural model is very accurate at a price of larger computational times.

Table 2. Timing comparison of BNM, T&B and GA for small-scale problems

n	P	Avg. Time (s)		
		BNM	T&B	GA
50	5	0.63	**0.40**	0.84
50	10	0.81	**0.73**	1.53
50	20	0.95	**0.81**	2.10
75	5	0.74	**0.68**	1.01
75	10	1.00	**0.87**	2.10
75	20	0.93	**0.91**	3.11
100	5	0.81	**0.75**	2.02
100	10	1.20	**1.02**	4.10
100	20	2.15	**1.94**	4.23

6 Conclusions

In this paper, we have demonstrated that neural network techniques can compete effectively with more traditional solutions and other heuristics to solve clustering problems. We have also presented a bidirectional neural model (BNM), which, unlike existing recurrent neural approaches, can be guaranteed a feasible solution to the problem. A new competitive dynamics has been proposed to the neural model ensuring the feasibility of solutions, and the convergence of BNM with the proposed computational dynamics has been demonstrated.

Other heuristics applied to the clustering problems involve somewhat arcane analogies from the physical or biotic world, where many unintuitive parameters such as temperature, long and short term memory, or pheromone characteristics, must be devised, estimated and applied. Here, we do not need determine anything; we need only decide the number of patterns (n), and the number of clusters (p). Moreover, the typically tuning problem of recurrent neural networks has been avoided using a competitive dynamics.

In this work we have analyzed a bidirectional neural model (BNM) and compared against several other heuristics like a GA and T&B. The experimental results show that our neural model is able of reporting better results than the rest heuristics. The fact that a simple neural model is able of computing that accurate solutions along with its simplicity can be a step towards making the research community of the potential inside neural networks. This is an important point, since most authors never actually use algorithms reporting high accuracy due to they are difficult to implement or understand.

References

[1] Alp, O., Erkut, E. and Drezner, Z. (2003). An efficient genetic algorithm for the p-median problem. *Annals of Operations Research* 122, 21-42.

[2] Bai-Ling, Z., Bing-Zheng, X. and Chung-Ping, K. (1993). Performance analysis of the bidirectional associative memory and an improved model from the matched-filtering viewpoint. *IEEE Transactions on Neural Networks* 4(5), 864-872.

[3] Bartezzaghi, E. and Colorni, A. (1981). A search tree algorithm for plant location problems. *European Journal of Operational Research* 7, 371-379.

[4] Christofides, N. and Beasley, J.E. (1982). A tree search algorithm for the problem p-mediates. *European Journal of Operational Research* 10, 196-204.

[5] Densham, P. and Rusthon, G. (1992). A more efficient heuristic for solving large p-median problems. *Paper in Regional Science* 71, 207-239.

[6] Drezner, Z. and Guyse, J. (1999). Application of decision analysis to the Weber facility location problem. *European Journal of Operational Research* 116, 69-79.

[7] Drezner, Z. and Hamacher, H.W. (2003). *Facility location: applications and theory.* Springer

[8] Durbin, R. and Willshaw, D. (1987). An analogue approach to the travelling salesman problem using an elastic net method. *Nature* 326, 689-691.

[9] Favata, F. and Walker, R. (1991). A study of the application of the Kohonen-type neural networks to the travelling salesman problem. *Biological Cybernetics* 64, 463-468.

[10] Galvao, R.D. (1980). A dual-bounded algorithm for the problem p-mediates. *Operations Research* 28, 1112-1121.

[11] Gee, A.H. and Prager, R.W. (1995). Limitations of neural networks for solving traveling salesman problems. *IEEE Transactions on Neural Networks* 6(1), 280-282.

[12] Hanjoul, P. and Peelers, D. (1985). A comparison of two dual-based procedures for solving the p-median problem. *European Journal of Operational Research* 20, 386-396.

[13] Hansen, P. and Mladenovic, N. (1997). Variable neighborhood search for the p-median problem. *Location Science* 5(4), 141-152.

[14] Hopfield, J. and Tank, D. (1985). Neural computation of decisions in optimization problems. *Biological Cybernetics* 52, 141-152.

[15] Hribar, M. and Daskin, M.S. (1997). A dynamic programming heuristic for the problem p-mediates. *European Journal of Operational Research* 101, 499-508.

[16] Hung, D.L. and Wang, J. (2003). Digital hardware realization of a recurrent neural network for solving the assignment problem. *Neurocomputing* 51, 447-461.

[17] Kariv, O. and Hakimi, S.L. (1979). An Algorithmic Approach to Network Location Problem. Part 2: The p-Median. *SIAM J. Appl. Math.* 37, 539-560.

[18] Khumawala, B.M. (1972). An efficient branch and bound algorithm for the warehouse location problem. *Management Science* 18(12), 718-731.

[19] Kosko, B. (1988). Bidirectional associative memories. *IEEE Transactions on Systems, Man and Cybernetics* 18(1), 49-60.

[20] Kung, S. (1993). *Digital neural networks.* New Jersey (USA), Prentice-Hall

[21] Rolland, E., Schilling, D. and J.R., C. (1997). An efficient tabu search procedure for the p-median problem. *European Journal of Operational Research* 96, 329-342.

[22] Rosing, K.E. and ReVelle, C.S. (1997). Heuristic concentration: two stage solution construction. *European Journal of Operational Research* 97, 75-86.

[23] Shams, S. and Gaudiot, J.-L. (1995). Implementing regularly structured neural networks on the DREAM machine. *IEEE Transactions on Neural Networks* 6(2), 407-421.

[24] Smith, K., Palaniswani, M. and Krishnamoorthy, M. (1998). Neural techniques for combinatorial optimization with applications. *IEEE Transactions on Neural Networks* 9(6), 1301-1318.
[25] Smith, K. (1996). An argument for abandoning the travelling salesman problem as a neural-network benchmark. *IEEE Transactions on Neural Networks* 7(6), 1542-1544.
[26] Teitz, M.B. and Bart, P. (1968). Heuristic methods for estimating the generalized vertex median of a weighted graph. *Operations Research* 16, 955-961.

Reducing the Complexity of Kernel Machines with Neural Growing Gas in Feature Space

Luigina D'Amato[1], José Ali Moreno[1], and Rosa Mujica[1,2]

[1] Laboratorio de Computación Emergente, Facultades de Ciencias e Ingeniería, Universidad Central de Venezuela, Caracas, Venezuela
[2] Postgrado de Física, Facultad de Ciencias, Universidad Central de Venezuela, Caracas, Venezuela
jose@neurona.ciens.ucv.ve
http://neurona.ciens.ucv.ve/laboratorio

Abstract. We present a method for complexity reduction of kernel non-linear classifiers by a sparsity process applied on the training data set. This is achieved through the neural growing gas clustering method in feature space. The kernel "trick" is used to extract a relevant data set into the feature space according to a geometrical consideration. Classical algorithms, SMO and Adatron learning, are then applied on the selected data to obtain the kernel machine solution. The approach covers a wide range of algorithms and improves current methods of sparse kernel based classification in two aspects. First, reduces the complexity of the training phase by extracting from the training data set a small representative subset of training instances. Second, the kernel machine obtained as solution is extremely sparse without any reduction of its generalization capacity. The method is experimentally compared using synthetic and real data benchmarks, with results of other proposed sparse kernel based classification methods.

Keywords: Learning Machines, Neural Networks.

1 Introduction

Kernel machines have become a standard tool in the arsenal of Machine Learning practitioners when handling regression and classification problems. Classically, the first approach to face these problems consisted in the use of linear methods, i.e. linear machines. Their elementary mathematical form allowed the development of simple training algorithms and the detailed study of their properties [1]. Without doubt one of the most successful concepts in this respect has been that of the maximal margin classifier also called perceptron of maximal stability. The Support Vector Machine (SVM) approach, based on the concept of Structural Risk Minimization [2] formalized all these methods into a very robust methodology. Of particular interest is the way in which these initially linear methods were generalized to non-linear decision rules using the Reproducing Kernel concept.

This idea commonly known as the kernel "trick" has been used extensively in generating non-linear versions of conventional linear algorithms [1,3,4]. The procedure works as follows: if the training examples enter in the algorithm to be generalized only in the form of dot products $\langle \mathbf{x} \cdot \mathbf{y} \rangle$, then they can be replaced by a kernel function $K(\mathbf{x}, \mathbf{y})$.

A symmetric function $K(\mathbf{x}, \mathbf{y})$ is a kernel if it fulfills Mercer's condition, i.e. the function K is (semi) positive definite. When this is the case there exists a mapping ϕ such that it is possible to write $K(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}) \cdot \phi(\mathbf{y}) \rangle$. The kernel represents a dot product on a different space F called feature space into which the original vectors are mapped. In this way a kernel function defines an embedding of examples into (high or infinite dimensional) feature vectors and allows the algorithm to be carried out in feature space without ever explicitly representing it. In the present work the "kernel trick" is applied to the growing neural gas [5] in order to obtain a general adaptive clustering method in feature space.

A noteworthy advantage that kernel machine methods frequently show is that of yielding sparse solutions. By sparseness we mean that the final classifier can be written in terms of a relative small number of input examples called support vectors (SVs). Besides the practical advantage of this property there are generalization bounds that depend on the sparseness of the resulting classifier [1]. However, the solutions provided by the training methods of kernel machines, in particular SVMs, have been empirically shown to be not maximally sparse [6–10]. It is observed for example, that the application of different training approaches of the same kernel machine over identical training sets render disctint sparsity. These facts have recently spawned much work on the sparseness issue of kernel machine solutions. It is also clear that the study of this subject is closely related to the efficiency issue of the training algorithms.

In the literature [6–11] three main approaches for achieving sparseness of the kernel machine solution can be found. A post-processing approach [7] that consists on solving the kernel machine problem using standard methods with the entire training set and then simplify the solution by eliminating linear dependencies of the SVs in feature space. An online approach [10] consisting on the modification of the training algorithm by the incorporation of a sparsification method in order to obtain an approximate probable convergent algorithm. Since the complexity of SV learning is super-linear in the number of samples, performing aggressive sparsification concurrent with learning has important impacts both on the sparseness as on the efficiency issues. A preprocessing approach [8,9,11] consisting on the application over the training set of a data selection procedure in order to extract a relevant subset of training examples. This data extracting procedure can be carried out in input space or in feature space taking advantage of the kernel trick. In the former case the kernel machine is trained with a smaller subset of randomly selected examples but conserving in the training algorithm the restrictions imposed by the total set of training examples. In the latter case the data extraction proceeds in feature space in such a way that the subset of extracted data forms a base of the subspace defined by the training examples. The extracted data subset captures the structure of the entire data space in feature space. The experiments reported in all these approaches indicate that the proposed sparsification procedures effectively reduce the number of SVs without affecting in relevant manner the generalization capacity of the final classifier. In order to elucidate the possible advantages of one method over the other controlled experiments on identical data benchmarks must be carried out. A priori an important issue in a discussion of the performance of the different sparsification methodologies is the computing cost involved. In this sense, the a *posteriori* approach, if admittedly produces a satisfactory sparse kernel machine solution, has the disadvantage of requiring the complete training data with the well known complexity drawbacks linked to training large data sets. The online approach makes some advances in the correct direction but results in a rather involved training algorithm. A fact that collides with the well known experimental practice of using the simplest, most efficient and surest available algorithm. We are of the opinion that the preprocessing approach is the most adequate since the reduction of the cardinality of the training set clears the way for the application of well known efficient training algorithms [12, 13] without significant complexity overheads. It is also found that for this case the resulting kernel machines are of sparsity comparable to those of the a *posteriori* approach.

In this paper we propose a non linear generalization of the growing neural gas as data extraction procedure. The neural gas clustering algorithm is used to extract the relevant training vectors into the feature space F taking advantage of the kernel theory and preserving the geometrical structure of the data into F. In a second step efficient training algorithms are applied to the subset of relevant training examples to obtain the final sparse kernel machine solutions. Experimental comparisons of the SMO [12] and Adatron [13] training rules over synthetic and real data benchmarks, with other sparse kernel based classification methods are carried out. The remainder of the paper is organized as follows: In section 2 we briefly overview kernel machine classification along with the SMO and Adatron training procedures used in the experiments. In section 3 the growing neural gas algorithm and its non linear generalization for training data selection is presented. In section 4 the experiments with applications of the method are presented and the conclusions are discussed in section 5.

2 Kernel Machine Classification

Consider a training set $\mathbf{S} = \{(\mathbf{x_1}, y_1), \cdots, (\mathbf{x_n}, y_n)\}$, $\mathbf{x_i} \in \mathrm{IR}^d$ and $y_i \in \{+1, -1\}$ where the $\mathbf{x_i}$ are called the training vectors in input space and the y_i are their corresponding class labels. The objective is to construct from the finite data set $\mathbf{S}$ a good decision function $f(\mathbf{x})$ such that

$$y_i = f(\mathbf{x_i}) \qquad \forall(\mathbf{x_i}, y_i) \in S \tag{1}$$

This problem has been exhaustively studied in the literature [1,13] and will not be repeated here but for completeness we recall the main steps and results.The criteria for a solution $f(\mathbf{x})$ of the classification problem to be acceptable is that the functional margins of all training points be positive:

$$\gamma_i = y_i f(\mathbf{x_i}) \geq 0 \qquad \forall(\mathbf{x_i}, y_i) \in S \tag{2}$$

In the case of a linear decision function this criterion can be extended to the geometrical margins

$$\Gamma_i = \frac{\gamma_i}{||\mathbf{w}||} = \frac{y_i\left(\langle \mathbf{w} \cdot \mathbf{x_i}\rangle + b\right)}{||\mathbf{w}||} \geq 0 \qquad \forall(\mathbf{x_i}, y_i) \in S \tag{3}$$

It is clear from relation (3) that the most stable solution should be the one with the greatest minimum geometrical margin. This is called the maximal margin classifier or perceptron with maximal stability. Since the definition of functional margin (2) is dependent on the scale of the weight vector, it is common practice to choose a scale such that the minimum functional margin is unity. With this assumption the problem of obtaining the parameters of the maximal margin classifier can be stated as

$$\begin{array}{ll} minimize & \frac{1}{2}||\mathbf{w}||^2 \\ s.t. & \gamma_i = y_i\left(\langle \mathbf{w} \cdot \mathbf{x_i}\rangle + b\right) \geq 1 \qquad \forall(\mathbf{x_i}, y_i) \in S \end{array} \tag{4}$$

There are two computational proposals for solving this optimization problem. In the SVM literature [1] there is the SMO algorithm [12] and in the Statistical Physics literature the Adatron algorithm [13]. The main difference between both proposals is that the SMO algorithm operates in batch mode, and the Adatron in sequential mode. The SMO requires all training samples to be given in advance, while in the Adatron a single training example is observed at each time and the update depends solely on that example. In the present work we compare results produced with both algorithms. As discussed above non linear generalizations of the maximal margin classifier can readily be done taking advantage of the kernel theory. Elaborations in this respect can be found in the literature [1,14]. Summarizing, it is shown that the form of the resulting kernel machine is

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i y_i K(\mathbf{x_i}, \mathbf{x}) + b \tag{5}$$

Where $K(\mathbf{x_i}, \mathbf{x})$ is the kernel function and the α_i are the Lagrange multipliers resulting from the solution of the optimization problem in its dual representation. In this work the Gaussian kernel is used throughout:

$$K(\mathbf{x_i}, \mathbf{x}) = exp\left(-||\mathbf{x_i} - \mathbf{x}||^2/2\sigma^2\right) \tag{6}$$

3 Neural Growing Gas in Feature Space

The neural gas type algorithms [5] are unsupervised learning algorithms that produce a sparse approximation of the distribution density of input points. The basic idea behind the neural gas approaches is to partition the input space into small regions, Voronoi polyhedric cells, in accordance with the distribution density of the sample of input examples considered. Each of these regions is defined by the neighborhood of a prototype or processing unit (P.U.) in its center. The distribution density of the P.U.s approximates the distribution density of the points in input space. In this way a compressed representation of the set of input examples which captures the structure of the entire data set is obtained. We take advantage of this property and generalize it with the kernel trick to obtain a data selection algorithm in feature space. In the growing version of the neural gas the population of prototypes increases during the learning phase guided by local error measures computed during the adaptive dynamics. The standard neural growing gas algorithm can be found in [5], the modifications introduced in this work refer mainly to the evaluation of the involved distances in feature space. The most important changes are the following:

1. When an input $\mathbf{x}$ is presented its distance to each of the prototypes $\mathbf{p_i}$, is computed in feature space:
$$||\mathbf{x_i} - \mathbf{p_i}||^2 = K(\mathbf{x},\mathbf{x}) + K(\mathbf{p_i},\mathbf{p_i}) - 2K(\mathbf{x},\mathbf{p_i}) \tag{7}$$
2. The local error of each prototype is hence computed in feature space.
3. Because the actual mapping from input space to feature space is not known the adaptation of the prototypes and of their topological neighbors is, as in the original algorithm, carried out in input space.
4. Once the desired set of prototypes is computed, each one of them is replaced by the nearest point in feature space, of the input data set. In this way the relevant points of the input data set are extracted.
5. The parameters σ, β, ε_b, ε_n, λ, governing the dynamics of the growing neural gas remain unchanged.

4 Experiments and Results

In this section the results of the application of the proposed method to the SMO and Adatron kernel machines learning algorithms on six binary classification data sets are presented.

4.1 Benchmark Datasets

Two synthetic datasets: the checkerboard problem [15] and a self-designed triangle board pattern presented in Figure 1 in the form of two 256x256 pixels images for a total of 65536 x 2 (components) data points. These synthetic datasets are chosen in order to depict graphically the effectiveness of the proposed method using several cardinalities of reduced training sets, extracted from an initial 2000 point data set selected at random. The latter results are compared with results obtained with the same number of points selected at random.

Four benchmark databases from the University of California (UC) Irvine repository [16]: ionosphere with 351 x 34 points, pima-indians-diabetes with 768 x 8 examples, tic-tac-toe with 958 x 9 samples, and mushroom with 8124 x 22 data points. All training data was linearly scaled to be in the [1,-1] range.

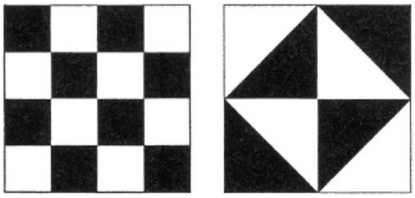


Fig. 1. Synthetic data benchmarks 256x256 pixels images. Checkerboard (CB) left, 16 black and white squares; Triangles-pattern (TP) right, 8 black and white triangles

4.2 Implementation Issues

The algorithms (Neural growing gas, SMO and Adatron) were implemented in the C language and compiled and run on a Linux environment with a AMD Athlon XP2600+ CPU. The Gaussian kernel function was used in all numerical tests. The parameters of the Gaussian and of the neural growing gas algorithm are selected on a trial and error basis over each data set. Once established, the same parameter set is used on both training algorithms. A tricky issue in the data extraction phase is the selection of the number of relevant training examples (prototypes) to be extracted with the neural gas from the initial training data set. This depended on the experimental situation, in particular on how large the problem was. The numbers used varied in the range from N/3 to N/2 with N the initial training set's cardinality.

4.3 Experimental Issues

The kernel machine training experiments were carried out in the following way:

- Each experiment was repeated 10 times with a fixed value of the kernel parameter.
- For each benchmark data set an initial training set of cardinality N was selected at random. The training set's cardinality on the synthetic benchmarks was N=2000. The UCI repository data set's cardinality was fixed at the values indicated in each database: Ionosphere N=151; pima-indians N=192; tic-tac-toe N=287; mushroom N=5687.
- The remaining data examples in the dataset were used for cross-validation.
- Experiments with and without the application of the neural gas data extracting procedure were compared. In the latter case the entire initial training data set was used. When the data extracting procedure is applied, the neural growing gas acts over the initial training data set and runs until the desired number of prototypes is reached. This set of prototypes is then used as reduced training set on both SMO and Adatron algorithms.
- The stopping criteria for the learning algorithms were the verification of the KKT conditions for the SMO and the convergence to a minimum margin of 0.999 for the Adatron.

The criteria for evaluating the performance of these methods is on one side the measured accuracy rate [11] and on the other their sparseness. The accuracy is measured by cross-validation of the trained kernel machine. The validation was carried out in two ways:

- Total validation (TVal): with the entire validation data set, a total average accuracy rate is measured.
- Random validation (RVal): selecting at random 100 samples from the validation set and calculating the accuracy rate. This validation was repeated 30 times and the average accuracy rate measured.

4.4 Synthetic Data Experimental Results

The first experiment consisted in training a Gaussian kernel machine ($\sigma_{gauss} = 0.6$) over data sets with 200, 400, 600 and 800 randomly selected examples from the CB and TP images. No data extraction was performed. The resulting accuracy rates are shown in the following table:

Table 1. Accuracy rate for experiments without data extraction on synthetic data. N=Training data set's cardinality, TVal=Total Validation, RVal=Random Validation, CB=Checkerboard and TP=Triangles-pattern

	Randomly Selected Training Examples $\sigma_{gauss} = 0.6$							
Algorithm -	N=200		N=400		N=600		N=800	
Data	% TVal	% RVal	% TVal	% RVal	% TVal	% RVal	% TVal	% RVal
SMO-CB	89.38	89.13	92.65	92.44	95.25	94.80	95.52	95.48
Adatron-CB	89.57	89.26	92.46	92.60	94.50	95.00	96.23	96.41
SMO-TP	91.54	91.63	94.15	93.85	96.68	96.64	96.81	96.75
Adatron-TP	91.22	91.36	94.71	94.71	97.70	97.89	97.57	97.80

Both algorithms show more or less the same performance with no essential differences in the observed accuracy rates. The number of support vectors for the kernel machines in these experiments are in the range of 28-50 for the Adatron and 114-273 for the SMO. The Adatron produces very much sparse results than SMO. In the following experiments reduced training sets of cardinalities 600 and 800 were extracted from a sample of 2000 randomly selected points from the CB and TP images respectively. The parameters used in the growing neural gas algorithm for data extraction on the synthetic data sets were $\sigma = 0.5$; $\beta = 0.0005$; $\varepsilon_b = 0.06$; $\varepsilon_n = 0.0005$; $\lambda = 500$. The following table shows the measured accuracy rates results together with the resulting number of support vectors:

These results show the effectiveness of the growing gas data extraction method in reducing the complexity of the kernel machines. In particular the Adatron algorithm seems to be superior to the SMO method both in accuracy and sparseness. The Adatron results are superior to those obtained with RSVM in [8] where the authors report for the checkerboard benchmark two machines with 50 and 100 SVs and average accuracies of 96.70% and 97.55% respectively.

Table 2. Accuracy rate for experiments with data extraction on synthetic data. N=Reduced Training data set's cardinality, TVal=Total Validation, RVal=Random Validation, CB=Checkerboard, TP=Triangles-pattern and SVs=Support Vectors

	Neural Gas extracted Training Examples $\sigma_{gauss} = 0.6$					
Algorithm -	N=600			N=800		
Data	% TVal	% RVal	SVs	% TVal	% RVal	SVs
SMO-CB	96.28	95.80	242	96.98	96.90	272
Adatron-CB	97.70	97.89	37	98.91	99.12	45
SMO-TP	97.60	97.54	185	98.13	98.05	232
Adatron-TP	97.62	97.31	33	99.01	99.12	44

4.5 UCI Repository Experimental Results

Table 3 summarizes the results obtained on the four UC Irvine repository databases. The parameters used in the growing neural gas algorithm for data extraction were the following $\sigma = 0.5$; $\beta = 0.0005$; $\varepsilon_b = 0.06$; $\varepsilon_n = 0.0005$; $\lambda = 300$.

Table 3. Experimental Results UCI Repository data. TVal=Total Validation, RVal=Random Validation, N_{initial}=Initial Training data set's cardinality, N=Reduced Training data set's cardinality and SVs=Support Vectors

		Neural Gas extracted Training Examples					Complete Training Set N_{initial}		
	Algorithms	% TVal	% RVal	N_{initial}	N	SVs	% TVal	% RVal	SVs
Ionosphere	SMO	93.44	93.40	200	100	65	95.30	95.20	108
σ_{gauss}=3.0	Adatron	95.47	95.50	200	100	78	95.37	95.45	92
Pima-Indians	SMO	78.89	79.90	576	300	206	75.16	75.20	406
σ_{gauss}=0.6	Adatron	81.23	80.00	576	300	125	78.91	78.95	175
Tic Tac Toe	SMO	99.04	98.95	672	200	180	98.78	98.79	454
σ_{gauss}=3.0	Adatron	99.41	99.39	672	200	117	98.89	98.99	220
Mushroom	SMO	99.25	99.26	5687	200	141	100	100	1048
σ_{gauss}=3.0	Adatron	100	100	5687	200	74	100	100	144

This results on real data confirms the effectiveness of the neural growing gas data extraction method in reducing the complexity of the kernel machines. It can be also observed that in general the Adatron algorithm produces a kernel machine with increased sparseness and superior accuracy when compared to the SMO. Comparing these results to those obtained with RSVM it can be appreciated that the Adatron results are superior in accuracy to those reported in [8] but the RSVM machines are of reduced complexity, except in the mushroom dataset. The support vectors for both kinds of kernel machines (RSVM to Adatron) on these data sets are found to be: ionosphere (35 to 78); pima-indians (50 to 125); tic tac toe (96 to 117) and mushroom (215 to 74). These differences can be attributable to the fact that in the RSVM method the user chooses from the beginning of the algorithm the desired number of support vectors limiting the attainable accuracy. In the neural growing gas method the user chooses the number of training examples but not directly the number of support vectors.

5 Conclusions

We have proposed the application of the neural growing gas algorithm in feature space as a data extraction procedure for the sparsification of the training sets of kernel machines in order to reduce their complexity without affecting significantly their generalization capacity. The method was experimentally tested with kernel classification machines on synthetic and real data benchmarks with both Adatron and SMO learning. It has been experimentally shown that the growing gas algorithm is an effective data extraction procedure that produces a sparse training set that conserves the structure of the data in feature space. The performance of the machines trained with the reduced data sets show in general the same or better performance than those trained with the complete data set. This may be a consequence of a reduction of data overfitting with the extracted training sets of smaller cardinality. From a practical point of view the method reduces the complexity of the kernel machine both in its training phase by reducing the cardinality of the training set and in its operative phase by the reduction of the number of resulting support vectors. The experimental results show a superiority of the Adatron algorithm over SMO training. This is an interesting result since the Adatron algorithm is a very simple perceptron like sequential procedure. Nevertheless further experiments must be carried out in order to generalize this claim for bigger data sets. The results show that the performance of the proposed method is comparable to that of RSVM machines. Finally, although the experiments were carried out on classification problems the method can be straightforwardly applied to regression problems.

References

1. Cristianini, N., and Shawe-Taylor, J.: An Introduction to Support Vector Machines. Cambridge University Press, Cambridge, England. (1982)
2. Vapnik,V. N.: Statistical Learning Theory. Wiley Interscience, New York. (1998)
3. Herbrich, R.: Learning Kernel Classifiers. The MIT Press, Cambridge,MA. (2002)
4. Sh$\ddot{o}$lkopf, B., Smola, A.: Lerning with Kernels. The MIT Press, Cambridge,MA. (2002)
5. Fritzke, B.: Some competitive learning methods. Technical Report from the Systems Biophysics Institute for Neural Computation, Ruhr Universitaet Bochum. (1997)
6. Burges, C. J. C., and Sh$\ddot{o}$lkopf, B.: Improving the accuracy and speed of support vector machines. In Advances in Neural Information Processing Systems, The MIT Press **9** (1997)
7. Downs, T., Gates, K., and Masters, A.: Exact simplification of support vectors solutions. Journal of Machine Learning Research 2: (2001) 293–297

8. Lee, Y., and Mangasarian, O. L.: RSVM: Reduced Support Vector Machines. In Proc. 1st SIAM Int. Conf. Data Mining 2001
9. Baudat, G., and Anouar, F.: Kernel-based Methods and Function Approximation. www.esat.kuleuven.ac.be/˜kpelckma/research/research/kernel/baudat2001.pdf
10. Engel, Y., Mannor, S., and Meir R.: Sparse Online Greedy Support Vector Regression. www-ee.technion.ac.il/˜rmeir/Publications/Ecml02Sogsvr.pdf
11. Lin, K.-M., and LIN, C.-J.: A Study on Reduced Support Vector Machines. IEEE Trans. Neural Networks, vol 14, (2003) 1449–1459.
12. Platt, J.: Fast training of Support Vector Machines using sequential minimal optimization. In Advances in Kernel Methods Support Vector Learning, The MIT Press, (1999) 42–65
13. Anlauf, J. K., and Biehl, M.: The Adatron: an adaptive perceptron algorithm. Europhysics Letters, **10** (1989) 687–692
14. Friess, T., Cristianini, N., and Campbell C.: The Kernel-Adatron Algorithm: a Fast and simple Learning Procedure for Support Vector Machines. In Shavlik, J., ed., Machine Learning: Proceedings of the Fifteenth International Conference, Morgan Kaufmann Publishers, San Francisco, CA, (1998)
15. Ho, T. K., and Kleinberg E. M.: Building projectable classifiers of arbitrary complexity. In Proceedings of the 13th International Conference on Pattern Recognition, Vienna, Austria, (1966) 880–885
16. Murphy, P. M., and Aha, D. W.: UCI repository of machine learning databases 1992. www.ics.uci.edu/~mlearn/MLRepository.html

Multirecombined Evolutionary Algorithm Inspired in the Selfish Gene Theory to Face the Weighted Tardiness Scheduling Problem

A. Villagra, M. De San Pedro, M. Lasso, and D. Pandolfi

Proyecto UNPA-29/B064, División Tecnología,
Unidad Académica Caleta Olivia,
Universidad Nacional de Patagonia Austral, Ruta 3 Acceso Norte s/n,
(9011) Caleta Olivia - Santa Cruz - Argentina
Phone: +54 297 4854888 ext.122
Fax: +54 297 4854888 ext.105
{avillagra, edesanpedro, mlasso, dpandolfi}@uaco.unpa.edu.ar

Abstract. In a production system it is usual to stress minimum tardiness to achieve higher client satisfaction. According to the client relevance, job processing cost, and many other considerations a weight is assigned to each job. An important and non-trivial objective is to minimize weighted tardiness. Evolutionary Algorithms have been successfully applied to solve scheduling problems. MCMP-SRI (Multiple Crossover Multiple Parents - Stud Random Immigrants) is a MCMP variant which considers the inclusion of a stud-breeding individual in a parent's pool of random immigrants. The Selfish Gene Algorithm proposed by Corno et al. is an interpretation of Darwinian theory given by the biologist Richard Dawkins. In this work we are showing a new algorithm that combines the MCMP-SRI and Selfish Gene approaches. This algorithm is used to face the weighted tardiness problem in a single machine environment. The paper summarizes implementation details and discusses its performance for a set of problem instances taken from the OR-Library.

Keywords: Evolutionary Algorithm, Multirecombination, Selfish Gene, Weighted Tardiness, Scheduling Problem.

1 Introduction

To achieve higher customer satisfaction current trends in manufacturing are focused today on production policies, which emphasize minimum weighted tardiness. Under this approach jobs to be delivered in a production system are usually weighed according to client's requirements and job relevance. Among other heuristics [6], [3], [27], [35], evolutionary algorithms (EAs) have been successfully applied to solve scheduling problems [2], [1], [5], [37], [38], [39].

Multiple Crossovers on a pool of Multiple Parents conformed by a Stud (breeding individual) and Random Immigrants (*MCMP-SRI*) is one of the latest variants of multirecombination [33], [34]. This method was applied in different scheduling problems for various single machine objectives for static and dynamic cases and the results obtained were satisfactory. In static problems, MCMP-SRI was applied to face Earliness and Tardiness [31], Weighted Tardiness [16], Average Tardiness [30] and Weighted Number of Tardy Jobs [17] objectives. In dynamic problems MCMP-SRI was applied to solve adaptability problems for Earliness and Tardiness objective [32], in partial and total dynamics problems for Weighted Tardiness [25] and Average Tardiness [15] objectives.

In 1976, the biologist Richard Dawkins published "The selfish gene" [12] a book, in which the theses of socio-biologist previously sated by E. O. Wilson in its "Socio-biologist" 1975 [40] were revealed. Dawkins' purpose is to examine the altruism and selfishness biology. He demonstrates that the more important factor in the evolution is not goodness of the species or group, as traditionally understood, but goodness of the individual or gene. For him and his followers, individuals are not more than machines created by genes for their survival. Paraphrasing Butler, "the hen is not more than an invention of the egg to be able to produce more eggs."

The Selfish Gene algorithm is an evolutionary algorithm based on this unorthodox Darwinian theory [7], [8]. This algorithm was recently applied in the field of electronic CAD, for determining the logic for a BIST (Built-In Self-Test) architecture based on Cellular Automata [9], [10], [11].

2 The Single Machine Weighted Tardiness Scheduling Problem

The single-machine total weighted tardiness problem [27], [35] can be stated as follows: n jobs are to be processed without interruption on a single machine that can handle no more than one job at a time. Job j ($j = 1,...,n$) becomes available for processing at zero time, it requires an uninterrupted positive processing time p_j on the machine, has a positive weight w_j, and a due date d_j by which it should ideally be finished. For a given processing order of the jobs, the earliest completion time C_j and the tardiness $T_j = \max\{C_j - d_j, 0\}$ of job j can readily be computed. The problem is to find a processing order of the jobs with minimum total weighted tardiness.

$$\sum_{j=1}^{n} w_j T_j$$

This model leads to an optimization problem that is NP-Hard [27], even with this simple formulation.

3 Selfish Gene Algorithms

In traditional genetic algorithms, a population is a group of coded solutions (individuals) in which each one is given a measurement of adaptation to the environment (problem) called fitness. The individuals of a population are selected to produce new individuals through a reproduction mechanism, known as crossover. So, those individuals that have better fitness than others have a high probability of transmitting their genes to the following generations.

Unlike traditional genetic algorithm the Selfish Gene Algorithm (SG) proposed by Corno et al. [7], [8] to solve “Knapsack 0/1 Problem”, the individuals are not important. The population is not an enumeration of individuals but an abstract model called Virtual Population (VP) that represents a statistical model of a pool of genes. The individuals don’t exist but they are represented by their chromosome to evaluate their adaptation to the problem. In a chromosome the position that a gene occupies is called the *loci,* while the value appearing at that position is the *allele.* The VP represents all the vectors of probabilities, where p_{ij} is the probability that an allele a_{ij} is selected. From VP any solution (chromosome) can be built however, some chromosomes are more likely to be built than others. The pseudo code shows the general structure of the SG algorithm. In the algorithm it is observed that the individuals are built from the VP through a selection function. This function very occasionally can generate a random individual, known as mutation process. In the implementation of Corno et al., reproduction mechanisms (crossover) don’t exist. The two generated individuals participate in a tournament where the quality of each solution is assessed. The winning solution increases the probabilities for its coded solution in the chromosome to be chosen, while the loser inversely diminishes them. The process finished when the probability vectors of the VP tend to stabilize.

```
SG Algorithm  ()
{        genome B, G1, G2;
         initialize all probabilities $p_{ij}$ to $1/n_i$ ;
         B = select_inidividual() ; /* best so far */
         do {     $G_1$ = select_individual();
           $G_2$ = select_individual();
           /* tournament */
           if (fitness  $G_1$) > (fitness  $G_2$)
           {
               reward_alleles ($G_1$);
               penalize_alleles ($G_2$);
               if (fitness  $G_1$) > (fitness B ) B = $G_1$;
           } else  {
               reward_alleles ($G_2$);
               penalize_alleles ($G_1$):
               if (fitness  $G_2$ ) > (fitness B ) B = $G_2$;
            }
             } while (steady_state () == FALSE)
         return B; }
```

4 MCMP-SRI-SG: Stud and Random Immigrants Combined with Selfish Gene Approaches

Multiple Crossovers per Couple (MCMP) [19] and Multiple Crossover Multiple Parent (MCMP) [21] are multirecombination methods which enhance the performance of traditional evolutionary algorithms, reinforcing the balance between exploration and exploitation of the search process.

Extreme exploitation can lead to premature convergence and intense exploration can make the search ineffective [26]. To solve this conflict, new trends in evolutionary algorithms make use of multiparent [22], [23], [18], and multirecombinative approaches [19], [20], [21]. The latter is called, *multiple crossovers on multiple parents* (MCMP). MCMP-SRI [36] introduces the use of a breeding individual (stud) which repeatedly mates individuals that randomly immigrate to a mating pool. Under this approach the random immigrants incorporate exploration and the multi-mating operation with the stud incorporates exploitation to the search process. Here, the process for creating offspring is performed as Fig. 1 shows. From the old population (of few individuals) an individual, designated the stud, is selected by means of proportional selection. The number of n_2 parents in the mating pool is completed with randomly created individuals (random immigrants). The stud mates every other parent, the couples undergo crossover and $2*n_2$ offspring are created. The best of these $2*n_2$ offspring is inserted in a temporary children pool. The crossover operation is repeated n_1 times, for different cut points each time, until the children pool is completed. Finally, the best offspring created from n_2 parents and n_1 crossover is inserted in the new population. The process is repeated until the new population is completed.

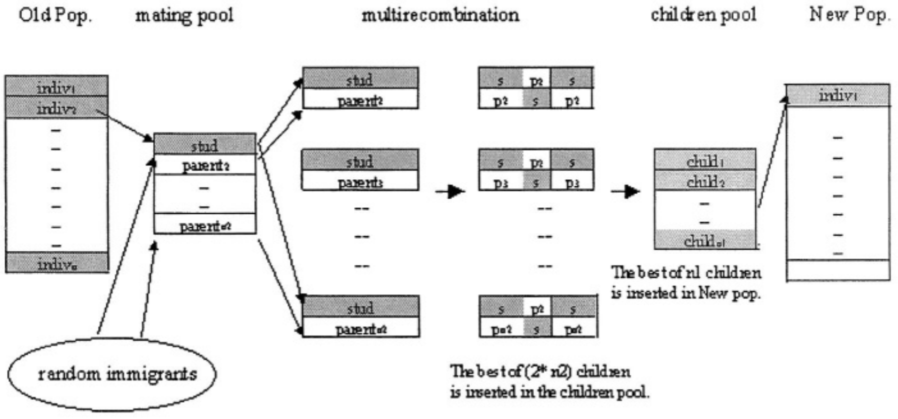


Fig. 1. Stud and random immigrants multirecombination process

MCMP-SRI-SG algorithm (see Fig. 2) is a variant of MCMP-SRI that applies the selfish gene theory substituting the population of individuals for a virtual population. The VP of stud contains vectors of accumulated frequencies of alleles, with which is possible to create a distribution of probabilities. Then the individuals known as stud can be generated with some mechanism of probabilistic selection such as proportional selection. Unlike the Selfish Gene algorithm, where the VP is updated by rewards and punishments for the winner and loser of a tournament, in this new algorithm only a reward to the winner of the process of mating generated by the stud and Random Immigrants is accumulated. This modification allows the generation of differential rewards, for example if the best offspring becomes the best in the evolution. For the case of random immigrants, all alleles are equally likely of being selected in any position of the chromosome. The MCMP-SRI-SG algorithm also differs from SG algorithm in that it applies crossover among the individuals of the mating pool with the objective of exploiting more promissory sub-schemes of the chromosome.

In this work we applied different crossovers which are adequate to permutation representation. The following well-known methods were used: PMX (Partial Mapped Crossover) Goldberg D. and Lingle R. [24], OX1 (Order Crossover 1) and OX2 (Order Crossover 2) Davis [13], [14], OSX (One Segment Crossover) Goldberg D. and Lingle R. [24], and CX (Cycle Crossover) Oliver [29].

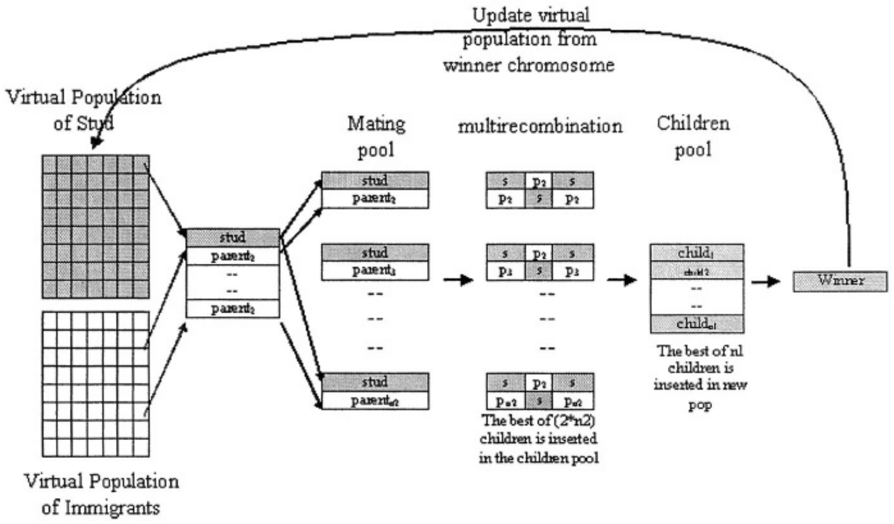

Update virtual population from winner chromosome
Virtual Population of Stud
Mating pool
multirecombination
Children pool
stud
parent
Winner
child
The best of n1 children is inserted in new pop
The best of (2*n2) children is inserted in the children pool
Virtual Population of Immigrants

Fig. 2. MCMP-SRI-SG Algorithm

5 Experiments and Results

The evolutionary algorithms were tested on instances selected from OR-library [4] for weighted tardiness scheduling problem. We performed a series of 20 runs for each of the selected instances of 40 jobs for weighted tardiness scheduling problems. Probabilities for crossover were set to 0.65 and for mutation (exchange) were set to 0.05, in all experiments. The number of $\mathbf{n}_1$ (number of crossover operations), and the number $\mathbf{n}_2$ (number of parents) were set to 16 and 18, respectively.

We studied the behavior of MCMP-SRI-SG algorithm in two stages. First, we compared this algorithm with MCMP-SRI algorithm for a few hard instances. Second, we analyzed MCMP-SRI-SG algorithm using different crossover operators.

To analyze the algorithm, the following relevant performance variables were chosen:

Ebest = ((best value - op_val) / op_val) 100. It is the percentile error of the best-found individual when compared with the known, or estimated (upper bound) optimum value op_val. It gives us a measurement on how far the individual is from that opt_val.

Mean Ebest = It is the mean value of Ebest throughout all runs.

Evals = Is the number of evaluations necessary to obtain the best-found individual in a run.

Mean Evals = It is the mean value of Evals throughout all runs.

Table 1 shows the results of a set of previous experiments realized with the purpose of comparing the performance of both algorithms. The table summarizes mean *Ebest* and mean *Evals* (in million of evaluations) values for MCMP-SRI and MCMP-SRI-SG algorithms. We can see that MCMP-SRI-SG obtained the best results for both performance variables except in the instance number 56. This instance (56) was especially hard for this problem. As the MCMP-SRI-SG obtains better results than MCMP-SRI and for space constraints in tables 2 and 3 we only display the results obtained with this algorithm.

Table 1. Result for MCMP-SRI and MCMP-SRI-SG

		MCMP-SRI		**MCMP-SRI-SG**	
Inst	**Upper Bound**	**Mean Ebest**	**Mean Evals**	**Mean Ebest**	**Mean Evals**
19	77122	0.31	2.854	***0.21***	***2.698***
41	57640	0.30	3.058	***0.08***	***2.351***
46	64451	***0.01***	3.100	***0.01***	***3.681***
56	2099	***6.10***	***1.151***	7.69	1.174
116	46770	0.58	3.978	***0.01***	***3.888***
	Avg	**1.46**	**2.828**	**1,60**	**2.758**

Table 2 summarizes mean Ebest values through all instances and operators for MCMP-SRI-SG algorithm. Results show that on average, the percentile error of the best found individual when compared with the best known objective value is the smallest (0.43%) under OX2 and greatest (1.34 %) under CX. We can see that OX, PMX, and OSX are the best values (in that order), and that maximum percentile error was for 56 instance for all crossovers.

Table 2. Mean Ebest values for the performance under MCMP-SRI-SG

Inst	PMX	OX1	OX2	CX	OSX	Inst	PMX	OX1	OX2	CX	OSX
1	2.34	2.81	1.22	5.08	3.20	**56**	8.13	8.84	7.69	9.03	11.98
6	0.00	0.06	0.00	0.00	1.26	**66**	0.01	1.16	0.02	0.76	0.34
11	0.00	0.51	0.06	0.28	0.07	**71**	0.00	0.40	0.01	0.39	0.03
19	0.34	0.49	0.21	0.58	0.36	**76**	0.00	0.00	0.00	0.00	0.00
21	0.00	0.10	0.00	0.23	0.00	**91**	0.24	2.01	0.08	1.58	0.78
26	0.00	0.00	0.00	6.67	0.00	**96**	0.00	0.12	0.00	0.28	0.04
31	0.00	1.44	0.00	2.64	0.00	**101**	0.00	0.00	0.00	0.00	0.00
41	0.03	2.50	0.08	1.14	0.31	**106**	0.00	0.00	0.00	0.00	0.00
46	0.05	0.23	0.01	0.29	0.01	**116**	0.15	2.11	0.01	1.30	0.97
51	0.00	0.00	0.00	0.00	0.00	**121**	0.06	0.50	0.02	0.28	0.07
						Avg	**0.57**	**1.08**	**0.43**	**1.34**	**0.85**
						Min	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
						Max	**8.13**	**8.84**	**7.69**	**9.03**	**12.0**

Table 3 summarizes mean Evals values, in millions of evaluations, through all instances and operators. Results show that on average, the minimum number of required to find the best individual is 1.990 under OX2, while the maximum is 3.344 under CX. We can see that OX, PMX, and OSX required least evaluations (in that order) for the average, minimum, and maximum values.

Table 3. Mean Evals values for the performance under MCMP-SRI-SG

Inst	PMX	OX1	OX2	CX	OSX	Inst	PMX	OX1	OX2	CX	OSX
1	0.982	0.990	1.731	1.558	1.870	**56**	0.967	2.006	1.174	2.613	2.822
6	1.268	2.835	0.967	3.189	2.308	**66**	3.365	4.613	3.818	4.948	4.287
11	1.265	3.005	1.943	4.106	2.663	**71**	3.651	4.658	3.370	4.825	4.566
19	3.302	4.660	2.698	4.912	3.618	**76**	0.039	0.055	0.037	0.091	0.058
21	3.367	5.004	2.847	4.937	3.849	**91**	4.519	4.822	3.455	4.753	4.697
26	0.225	0.304	0.139	0.806	0.684	**96**	3.455	4.879	2.674	4.911	4.629
31	2.048	3.028	1.036	2.903	1439	**101**	0.036	0.039	0.036	0.088	0.045
41	2.879	4.357	2.351	4.892	3.729	**106**	0.416	1.723	0.344	1.772	1.756
46	3.092	4.643	3.681	4.779	4.066	**116**	4.627	4.649	3.888	4.985	4.765
51	0.222	0.559	0.213	0.924	0969	**121**	3.635	4.901	3.401	4.889	4.392
						Avg	**2.168**	**3.087**	**1.990**	**3.344**	**2.861**
						Min	**0.036**	**0.039**	**0.036**	**0.088**	**0.045**
						Max	**4.627**	**5.004**	**3.888**	**4.985**	**4.765**

6 Conclusions

Evolutionary algorithms are robust search algorithms in the sense that they provide good solutions to a broad class of problems which otherwise are computationally intractable. To improve EAs performance, multrecombined EAs allow multiple interchange of genetic material among multiple parents (MCMP).

To ameliorate the search process, by means of a better balance between exploration and exploitation, the concept of stud and random immigrants was inserted in MCMP-SRI. The presence of the stud ensures the retention of good features of previous solutions and the random immigrants, as a continuous source of genetic diversity, avoid premature convergence.

The SG Algorithm is a novel optimization approach inspired in the biological Selfish Gene Theory, which the evolution of individuals' population is substituted by the evolution of a pool of genes and no recombination method is used.

The MCMP-SRI-SG is an interesting algorithm which combines both approaches. It showed a good performance by achieving existent benchmarks in weighted-tardiness problems. The good performance is comparable with MCMP-SRI particularly when use PMX and OX2 crossover.

In further work we will explore different optimization problems and find alternative ways of inserting knowledge in MCMP-SRI-SG.

Acknowledgements

Our acknowledgment to the cooperation and constructive critics provided by the research group and to the Universidad Nacional de la Patagonia Austral for their support to the research group.

Mainly and in his memory to Dr. Raúl Gallard who knew how to inspire in this group desires of enhancement and for whom we will keep the maximum affection, respect and gratefulness.

References

1. Burke E.K., Coewling P.I., Keuthen R.; Effective Heuristic and Metaheuristic Approaches to Optimize Component Placement in Printed Circuit Board Assembly; CEC2000, pp. 301-309, July 2000, La Jolla, USA.
2. Basseur M., Seynhaede F., El-ghazali; Design of Multiobjective Evolutionary Algorithms: Application to the Flow-Shop Scheduling Problem; CEC2002; pp. 1151-1156, May 2002, Honolulu, Hawai.
3. Chen T. and Gupta M.; Survey of scheduling research involving due date determination decision, European Journal of Operational Research, vol 38, pp. 156-166, 1989.
4. Beasley J. E. Weighted Tardiness Scheduling, OR Library, http:/ mscmga.ms.ic.ac.uk/
5. Cowling P., Graham K., Han L.; An Investigation of Hyper heuristic Genetic Algorithm Applied to a Trainer Scheduling Problem, CEC2002; pp. 1185-1190, May 2002, Honolulu, Hawai.
6. Crauwels H.A.J., Potts C.N. and Van Wassenhove L.N.; Local search heuristics for the single machine total weighted tardiness scheduling problem, Informs Journal on Computing 10, pp. 341-350, 1998.

7. Corno F., Sonza Reorda M., Squillero G.; The Selfish Gene Algorithm: a new Evolutionary Optimization Stategy; 13th Annual ACM Symposium Applied Computing, Atlanta, Georgia (USA) February 1998, pp. 349-255.
8. Corno F., Sonza Reorda M., Squillero G.; A new Evolutionary Algorithm Inspired by the Selfish Gene Theory ICEC98: IEEE International Conference on Evolutionary Computation May 1998 pp 575-580.
9. Corno F., Sonza Reorda M., Squillero G.; Exploiting the Selfish Gene Algorithm for Evolving Hardware Cellular Automata CEC2000: Congress on Evolutionary Computation, San Diego (USA), July 200o pp 1401-1406.
10. Corno F., Sonza Reorda M., Squillero G.; Exploiting the Selfish Gene Algorithm for Evolving Hardware Cellular Automata IJCNN2000: IEEE-INNS-ENNS International Joint Conference Neural Networks, Como (Italy), July 2000 pp 577-581.
11. Corno F., Sonza Reorda M., Squillero G.; Evolving Effective CA/CSTP BIST Architectures for Sequential Circuits SAC2001: ACM Symposium on Applied Computing, March 2001, Las Vegas (USA), pp 345-350.
12. Dawkins Richard; The selfish gene, Oxford University Press, 1976.
13. Davis L.; Applying adaptive algorithms to domains, in proceedings of the international Joint Conference on Artificial Intelligence, pp. 162-164, 1985.
14. Davis L.; Handbook of Genetic Algorithms, New York; Van Nostrand Reinhold Computer Library, 1991.
15. De San Pedro M.E., Lasso M., Villagra A., Pandolfi D., Gallard R.; Solutions to the Dynamic Average Tardiness Problem in the Single machine Environments; CACIC'03 IX Congreso Argentino de Ciencias de la Computación, La Plata, Argentina, octubre, 2003, pp. 1251-1258.
16. De San Pedro M.E., Pandolfi D., Villagra A., Vilanova G., Gallard R.; Stud and immigrants in multirecombined evolutionary algorithm to face weighted tardiness scheduling problems; CACIC'01 VII Congreso Argentino de Ciencias de la Computación, El Calafate, Argentina, octubre, 2001, pp. 1251-1258.
17. De San Pedro M.E., Villagra A., Lasso M., Pandolfi D., Diaz Vivar M., Gallard R.; CSITeA03 International Conference on Computer Science, Software Engineering Information Technology, E-Business and Applications, Rio de Janeiro, June 2003, Brazil, pp. 438-443.
18. Eiben A.E., Bäch Th.; An Empirical investigation of multi-parent recombination operators in evolution strategies. Evolutionary Computation, 5(3):347-365, 1997.
19. Esquivel S., Leiva A., Gallard R; Multiple Crossovers per Couple in Genetic Algorithms, Proceedings of 4th IEEE Conference Evolutionary Computation, ICEC97 Indianapolis. USA, April 1997, pp. 103-106.
20. Esquivel S., Leiva A., Gallard R; Couple Fitness Based Selection with Multiple Crossover per Couple in Genetic Algorithms, Proceedings of the International Symposium of Engineering of Intelligent Systems (EIS'98), La Laguna, Tenerife, Spain Vol. 1, pp 235-241.
21. Esquivel S., Leiva A., Gallard R; Multiple Crossover between Multiple Parents to improve search in Evolutionary Algorithms, Proceedings of Congress on Evolutionary Computation, CEC Washington DC, USA, 1999, pp. 1589-1594.
22. Eiben A.E., Raué P-E. and Ruttkay Zs; Genetic algorithms with multi-parent recombination, In Davidor, H-P. Schwefel and R. Männer editors, Proceeding of the 3rd Conference on Parallel Problem Solving from Nature, number 866 in LNCS, pp 78-87, Spring-Verlag, 1994.

23. Eiben A.E., van Kemenade C.H.M. and Kok J.N.; Orgy in the computer: Multi-parent reproduction in genetic algorithms, In F. Moran, A. Moreno, J.J. Merelo and P. Chacon, editors, Proceedings of the 3rd European Conference on Artificial Life, number 929 in LNAI, pages 934-945, Springer-Verlag, 1995.
24. Goldberg D. and Lingle R.; Alleles, loci and the traveling salesman problem, in Proceeding of the First International Conference on Genetic Algorithm, Lawrence Eribaum Associates, pp. 154-159, Hillsdale, NJ, 1987.
25. Lasso M., Pandolfi D., De San Pedro M.E., Villagra A., Gallard R.; Heuristics to Solve Dynamic W-T problems in Single Machine Environments; CSITeA03 International Conference on Computer Science, Software Engineering Information Technology, E-Business and Applications, Rio de Janeiro, June 2003, Brazil, pp. 432-437.
26. Michalewicz M.; Genetic Algorithms + Data Structures = Evolution Programs; Springer, third revised edition, 1996.
27. Morton T., Pentico D.; Heuristic scheduling systems; Wiley series in Engineering and technology management. John Wiley and Sons INC, 1993.
28. Madureira A., Ramos C., do Carmo Silva S.; A Coordination Mechanism for Real World Scheduling Problems Using Genetic Algorithms; CEC2002; pp. 175-180, May 2002, Honolulu, Hawai.
29. Oliver I. Smith D., Holland J., A study of permutation crossover operators on the traveling salesman problem, in European Journal o Operational Research, pp. 224-230,1986.
30. Pandolfi D., Lasso M., De San Pedro M.E., Villagra A., Gallard R.; Evolutionary Algorithms to solve average tardiness problems in the single machine environments; CSITeA03 International Conference on Computer Science, Software Engineering Information Technology, E-Business and Applications, Rio de Janeiro, June 2003, Brazil, pp. 444-449.
31. Pandolfi D., Vilanova G., De San Pedro M.E, Villagra A.; Gallard R.; Solving the single-machine common due date problem via studs and immigrants in evolutionary algorithms; Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics, Vol. III Emergent Computing and Virtual Engineering, pp. 409-413, Orlando, Florida July 2002.
32. Pandolfi D., Vilanova G., De San Pedro M.E, Villagra A.; Gallard R.; Adaptability of multirecombined evolutionary algorithms in the single-machine common due date problem; Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics, Vol. III Emergent Computing and Virtual Engineering, pp. 401-404, Orlando, Florida July 2002.
33. Pandolfi D.; De San Pedro M.E.; Vilanova G., Villagra A.; Gallard R. Multirecombining random and seed immigrants in evolutionary algorithms to solve W-T scheduling problems, Proceedings ACIS International Conference on Computer Science, Software Engineering, Information Technology eBusiness and Application Foz Iguazu Brazil 2002.
34. Pandolfi D., De San Pedro M.E., Villagra A., Vilanova G., Gallard R.; Studs Mating Immigrants in Evolutionary Algorithm to Solve the Earliness-Tardiness Scheduling Problem; In Cybernetics and Systems of Taylor and Francis Journal, (U.K.), pp. 391-400, June 2002.
35. Pinedo Michael; Scheduling: Theory, Algorithms and System, Prentice Hall, first edition, 1995, pp. 44--48,143-145.
36. Pandolfi D.; Vilanova G., De San Pedro M.E.; Villagra A.; Gallard R. Multirecombining studs and immigrants in evolutionary algorithm to face earliness-tardiness scheduling problems, Proceedings of the International Conference in Soft Computing University of Paisley, Scotland UK June 2001 pp. 138.

37. Reeves C.; A genetic algorithm for flow sequencing, Computers and Operations Reserach, vol 22, pp. 5-13, 1995.
38. Shaw K.J., Lee P.L., Nott H.P., Thompson M.; Genetic Algorithms for Multiobjective Scheduling of Combined Batch/Continuous process Plants, CEC2000, pp. 293-300, July 2000, La Jolla, USA.
39. Tsujimura Y., Gen M., Kubota E.; Flow shop scheduling with fuzzy processing time using genetic algorithms. The 11th Fuzzy Systems Symposium pp.248-252, Okinawa 1995.
40. Wilson E. O.; Sociobiology: the new synthesis, Harvard University Press, 1975.

A Novel Approach to Function Approximation: Adaptive Multimodule Regression Networks

Wonil Kim[1], Chuleui Hong[2], and Changduk Jung[3]

[1] College of Electronics and Information Engineering, Sejong University, Seoul, Korea
wikim@sejong.ac.kr
[2] Software School, Sangmyung University, Seoul, Korea
hongch@smu.ac.kr
[3] Seoul Information Communication Technology University, Seoul, Korea
jcd@sit.ac.kr

Abstract. In a function approximation task using neural network, there exist many cases in which the distributions of data are so complex that the regression with single network does not perform the given task well. Employing multiple modules that performs regression in different regions respectively may be a reasonable solution for this case. This paper proposes a new adaptive modular architecture for regression model, and simulates its performance on difficult problems that are not easily approximated using traditional neural network learning algorithms. Regression modules are added as learning proceeds, depending on the local distribution of data. The use of a local distribution that captures the underlying local structure of the data offers the advantage of adaptive regression.

1 Introduction

Given a complicated input-output mapping problem, it is hard to design a single network structure and solve the problem with this architecture. It usually takes a long time to train a monolithic network with many nodes and layers, and may not result in good generalization. In modular network architecture, each module is assigned to a specific (local) area and focuses only on its special area. Learning is more efficient when a neural network is organized in this way [1].. Modular networks utilize the principle of divide and conquer, which permits us to solve a complex computational task by dividing it into simpler subtasks and then combining their individual solutions. A modular approach also provides a better fit when the input-output mapping to be learned is discontinuous.

If a network employs an algorithm in which modules can be added adaptively according to the data space in training phase, then it can solve the *pre-knowledge* problem faced by most of the current modular systems. In general, it is unclear as to how we may modularize a network for a new task not encountered before, about which little heuristic information is available to construct. Unfortunately, most adaptive network construction algorithms, such as cascade correlation [2], insert nodes one at a time into a network, and do not introduce modules, since it is hard to delineate the purpose or scope of the module when it is introduced adaptively.

The use of a modular approach may also be justified on neurobiological grounds. Modularity appears to be an important principle in the architecture of vertebrate nervous systems, and there is much that can be gained from the study of learning in modular networks in different parts of the nervous system. The proposed algorithm in this paper trains one module after another, so one module may be trained extensively when the regression is hard and other modules can be trained using a small number of iterations when the regression is easy to learn. Each module is trained on a subset of the training data on which existing modules do not perform well, so that the total computational effort in training is much less than for traditional neural networks.

The structure of this paper is as follows. Chapter 2 and 3 discusses adaptive networks and modular approaches respectively. Chapter 4 proposes Adaptive Multimodule Regression Networks. Simulations and evaluation results of the proposed network is presented in Chapter 5. Chapter 6 concludes.

2 Adaptive Function Approximation

2.1 Nature of Adaptive Networks

Space is one fundamental dimension of the learning process; time is the other. The spatiotemporal nature of learning is exemplified in many learning tasks. Moreover, animals have an inherent capacity to represent the temporal structure of experience. Such a representation makes it possible for an animal to adapt its behavior to the temporal structure of an event in its behavioral space.

When a neural network operates in a stationary environment, the network can be trained by a supervised learning technique. Even in this stationary environment, the performance of supervised learning can be improved if the structure of a network can be decided adaptively during the network training session.

A neural network will undergo training sessions that modify the synaptic weights to match the environment. After the learning process, the network will capture the underlying statistical structure of the environment. Thus a learning system will rely on its memory to recall and exploit past experience. This process itself can be accomplished adaptively. Frequently, we do not have any information about the environment in advance, thus making the learning task hard to accomplish in a satisfactory manner. Instead of using a fixed number of modules, the proposed algorithms vary the number of regression modules during network training, adapting the whole network structure according to the given environment.

2.2 Adaptive Architectures in Neural Networks

For some problems that are easily learnable, the pruning approach may be a good candidate. This approach trains a large network initially and then repeatedly prune this network until size and performance of the network are satisfactory. Lecun, Denker, and Solla [3] proposed the *optimal brain damage* (OBD) method. In this method, by examining the second derivatives of the error function with respect to weights, they can identify which weights are to be pruned from the network. Hassibi and Stork [4] proposed the *optimal brain surgeon* (OBS) algorithm that is more general than OBD, but requires more extensive computations. But the pruning methods cannot capture the modular nature of a problem accurately. Pruning may lead to a network that represents the global data space well, but this approach suffers from spatial-cross talk problems for data sets with highly local features.

In most clustering networks, the number of output nodes is fixed. Carpenter and Grossberg [5] have developed networks called ART1, ART2 and ART3 that are adaptive to some extent. This approach assumes to have a finite maximum number of output units in advance, but does not use them until needed. At any given time there would be some output units in use, detecting the known categories, while others are waiting in the wings until needed. These networks overcome the stability-plasticity dilemma by accepting and adapting the stored prototype of a category only when the input is sufficiently similar to it. The main difference between ART and the proposed approach is that ART is a clustering network whereas the proposed networks performs regression, which is a much harder task.

Frean [6] proposed an algorithm for developing a multilayer hierarchical network. In this method, the network develops subnetworks for misclassified samples. At each step, subnets are added that operate exclusively on the misclassified samples. The number of layers in the resulting network is at most logarithmic in the number of training samples. But, this approach is suitable only to classification problems, not for function approximation problems.

Sirat and Nadal [7] also proposed a tree-structured network, but the nodes in the network resemble decision nodes in a decision tree instead of directly modifying the output of a higher level node. In this case, one node examines only the training samples for which the parent node output is 1, while a sibling node is used to make the best decision for the training samples for which the parent node output is 0. Like the Upstart algorithm, the Neural tree is implemented for classification problems, not function approximation problems.

Fahlman and Lebiere [2] proposed the *cascade correlation* algorithm that allows a network structure to add nodes adaptively according to the problem. Unlike the upstart algorithm, hidden nodes added at the outer layers, not the inner layer; unlike the neural tree, the final decision is made only at the outermost nodes, not at the leaves of a tree. However, the resulting network is unimodular, and performs poorly on problems where local specialization is needed, as observed by Lee [8].

3 Multimodule Approach

3.1 Spatial and Temporal Crosstalks

Jacobs mentioned that there are two kinds of problems which motivate us to use a modular architecture [1]. One is *spatial crosstalk* which occurs when output units of a network provides conflicting error information to a hidden unit. He noted that this occurs when the backpropagation algorithm is applied to a single network containing a hidden unit that projects to two or more output units. Plaut and Hinton [9] noted that if a network architecture has a separate module for each output unit, then it is immune to spatial crosstalk.

Another problem is that a unit might receive inconsistent training information at different times, which creates *temporal crosstalk.* For example, when a network is trained for one function, the network is specialized to that function. When the same network is trained for second (different) function which may be applicable in a different region of the data space, the network's performance for the first function will tend to be degraded. According to Sutton [10], backpropagation tends to modify the weights of hidden neurons that have already developed useful properties. Consequently, after being trained on the second task, the network is no longer able to perform the first task.

There are many more reasons why spatial and temporal crosstalk should be eliminated. The proposed adaptive modular networks are designed to tackle these problems, and successfully cope with both spatial and temporal crosstalk problems. It adaptively employs multiple modules for specific region, hence eliminates spatial crosstalk. Also it adds modules one by one depending on the performance of the learning task, hence eliminates the temporal crosstalk.

3.2 Modular Architectures in Neural Networks

Modular neural networks have been studied in biological neural computations because of the modular nature of the human brain. Such an approach has been used successfully for behavior control in robotics [11], phoneme classification [12], character recognition [13], task decomposition [14], and piecewise control strategy [115]. A modular network performs task decomposition in the sense that it learns to partition a task into two or more functionally independent tasks and allocates distinct network modules to learn each task. This property enhances the efficiency of learning a particular task. If there is a natural way of decomposing a complex function into a set of simpler functions, then a modular architecture should be able to learn the simpler functions faster than a single network can learn the undercomposed complex function. Another merit of modular networks is their generalizing ability. The modular architectures are capable of developing more suitable representations that those developed by non-modular networks.

Though modularity promotes effective learning in neural networks [1], it is not easy to determine how to modularize a network for a new task. Most adaptive algorithms [2,5] successively insert (into a network) nodes but not modules, since it is generally not possible to predetermine the purpose of a module. Most of the traditional modular algorithms require prior knowledge, and the number of modules for a given task is decided in advance. In such cases, the number of modules may be too many (implying excessive computational effort in training, and poor generalization properties) or too little (implying poor quality results).

The proposed algorithms use a reference vector for characterizing each module; each module is associated with one corresponding reference vector. In using the notion of reference vectors, whose relative proximity is used to determine the choice of the appropriate module, our algorithm bears an analogy with learning vector quantization (LVQ) [16] and counterpropagation [17] algorithms. However, those algorithms are not adaptive, not modular, and were developed for classification and pattern association tasks, hence not appropriate for regression problems. One of the advantages of the proposed approach is that the input space can be analyzed according to the input distribution without any prior knowledge. The clustering of an input distribution as well as function approximation (regression) on local regions are adaptively achieved.

4 The Proposed Adaptive Multimodule Regression Networks

4.1 The Localized Adaptive Multimodule Approximation Network (LAMAN)

Initially, each module is trained on a subset of the training data on which existing modules do not perform function approximation well. The algorithm consists of repeated introduction of new modules, until all the data are used. When a new module is introduced, it is trained on the remaining input data on which currently existing modules do not perform well enough.

Only one reference vector per module is used. Even though a LAMAN module's reference vector starts at a region boundary, it is updated during the training process to move to the centroid. The LAMAN algorithm also allows the size of the region (associated with each reference vector) to vary, resulting in a better representation if the data set has heterogeneous regions of inherently different sizes. It is called the Localized Adaptive Multimodule Approximation Network since it can capture the underlying local structure of the data adaptively. The main algorithm is as follows:

```
REPEAT
   1. Let $T_i'$ be the collection of $M_i$'s candidate input vectors $x_i'$ □ $T$ such that
        • $distance(x_i', R(i)) \le p1 \times \overline{D}_R(M_i, T_i)$, and
        • $error(M_i, x_i') \le p2 \times E_{old}(M_i, T_i)$,
      where $p1 > 0$, $p2 > 0$, $\overline{D}_R(M_i, T_i)$ is the average distance from the
      reference vector of $M_i$ to elements of $Ti$, $error$ is the squared error of $M_i$
      on $x_i'$, and $E_{old}(M_i, T_i)$ is the previous MSE of $M_i$ on $T_i$.
   2. Train $M_i$ on $T_i'$ for a predefined number of iterations, modifying the
      reference vector $R(i)$ so that it is located at the centroid of $T_i'$.
   3. Calculate $E_{new}(M_i, T_i')$ (the new MSE) and average current distance
      $\overline{D}_{Rnew}(M_i, T_i')$ from $R(i)$ to elements in $T_i'$.
UNTIL $E_{new}(M_i, T_i') \le \theta$ or $\frac{E_{old}(M_i, T_i)}{E_{new}(M_i, T_i')} \le p3$ or $curiteration \ge maxiteration$
 (where $\theta$, $p3$ and $maxiteration$ are predefined).
```

4.2 The Convex Hull Variation (CHAMAN)

Clusters are not symmetric in practical problems, making it difficult to use traditional competitive neural networks that use a Euclidean distance metric. The reference vector by itself may not adequately represent a region in data space. This section proposes a new approach to solve this problem.

The convex hull of a set *D* of points is the smallest convex set containing *D,* and provides a useful way of describing a cluster, and thus the set of points in the region associated with a module. When an input data region has a radially asymmetric shape, a module region can be described more accurately by the convex hull boundary. The Convex Hull Adaptive Multimodule Approximation Network (CHAMAN) algorithm represents each region using a reference vector (centroid) as well as the region's convex hull boundary. Points on the boundary of the convex hull are obtained during the module learning process.

The CHAMAN algorithm varies from the LAMAN algorithm in the adaptive training phase: a polygonal approximation for the convex hull boundary is computed for each region on which a module has been trained, using a fast incremental algorithm [18,19] that generates a set of points on the boundary of the convex hull, with adjacency information. A given input vector is assumed to be outside the convex hull if the distance between the reference vector and the input vector is greater than the distance between the reference vector and the point of intersection (p.i.c) of the convex hull boundary with the ray from the reference vector passing through the input vector, as shown in Fig. 1.

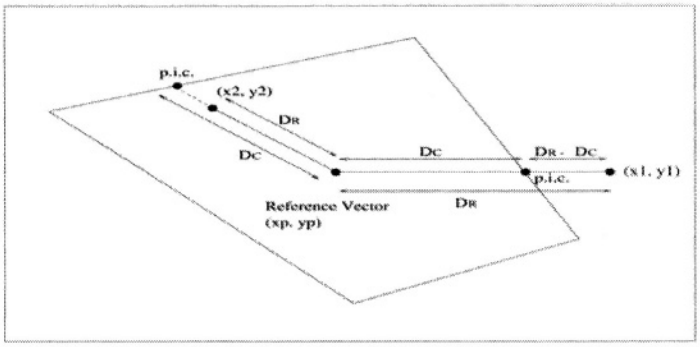


Fig. 1. Deciding whether an input vector is inside or outside convex hull

The network output is chosen as follows in the CHAMAN algorithm, where *DR(Mi, x)* denotes the distance between input vector *x* and a reference vector for module *Mi,* and *DC(Mi, x)* denotes the distance between that reference vector and the *p.i.c.* for *x*.

1. If *DR(Mi, x)* > *DC(Mi, x),* (i.e., *x* lies outside the convex hull) for every module, the network output is the output of the module with smallest *DR(Mi, x) - DC(Mi, x).*
2. Otherwise (i.e., *x* lies inside the convex hull), the network output is the output of the module with smallest *DR(Mi, x)* such that ***DR(Mi, x)* ≤ *DC(Mi, x)*.**

4.3 The Fuzzy Variations (FAMAN and FLAMAN)

We also explore the notion of fuzzy membership called Fuzzy Adaptive Muitimodule Approximation Networks (FAMAN) and Fuzzy Linear Adaptive Multimodule Approximation Network (FLAMAN), It is based on the relative position of an input data vector along a ray from the reference vector to the convex hull.

In some problems, clusters of data are not uniformly distributed over the corresponding region in the Voronoi space. For such problems, fuzzy membership grades (determined from the distribution of data in each convex hull) allow us to capture problem-specific variations in different regions of the input space. Fuzzy membership may be defined in several ways, of which the following is one possible definition, used in our experiments:

$$F(x, M_i) = \begin{cases} 1.0 & \text{if} \quad D_c(M_i, x) \geq C \times D_R(M_i, x) \\ e^{D_c(M_i,x) - C \times D_R(M_i,x)} & \text{otherwise} \end{cases}$$

Fuzzy membership functions have been proposed by other researchers for input space partitioning. For instance, Nomura and Miyoshi[24,25] use Gaussian membership functions that depend on the distance of an input vector from the mean and the width (ó) of the Gaussian. However, their work assumes that clusters are symmetric, with well-defined óvalues for each input dimension, whereas our algorithms are designed to be applicable to asymmetric clusters as well.

We have experimented with two variants of fuzzy module selection.

1. In the Fuzzy Adaptive Multi-module Approximation Network (FAMAN) algorithm, network output is the output of the modules with highest fuzzy membership value.
2. In the Fuzzy Linear Adaptive Multi-module Approximation Network (FLAMAN) algorithm, network output is the linear combination (weighted sum) of the outputs of relevant modules, where the weights are fuzzy membership grades.

5 Simulations and Evaluation

Experiments were conducted with 3 data sets, each with two input dimensions and one output dimension. Data Set 1 has three disjoint regions in the input space where different Gaussian functions are located. Data Set 2 contains five disjoint regions (one function in the middle and 4 surrounding functions) in the input space where different Gaussian functions are located. Data Set 3 contains five disjoint regions in the input space where five different Gaussian functions are located. For fair comparison, we employed two hidden nodes per module for the LAMAN, the CHAMAN, the FLAMN, and the FAMAN networks.

The monolithic network trained by backpropagation (BP) was equipped with 5 and 10 hidden nodes. Both monolithic networks are trained for 10000 seconds. The MSE results were averaged over 10 runs. In all cases, the LAMAN, the CHAMAN, the FAMAN and the FLAMAN outperformed BP networks by a factor of 50 or more in speed and 3-6 in MSE. Results on Data Sets 1-3, given in the Fig. 2–4, show that CHAMAN performs best in problems characterized by asymmetric data regions. The standard deviation of MSE for the CHAMAN algorithm was two orders of magnitude smaller than those for the unimodular BP networks.

For Data Set 3, more experiments were conducted with large unimodular BP networks with 100 and 500 hidden nodes. However, each BP networks produced MSE of 0.011707 and 0.006307 respectively after 10000 seconds, much poorer performance than the much smaller networks trained using the proposed algorithms in this paper.

The FAMAN algorithm performed at least as well as the CHAMAN algorithm on these data sets, but the current results are not conclusive in favor of a specific one of these variants. Overall, for radially asymmetric data sets, employing convex hull methods excelled over all other methods. Without such radial asymmetry, performance is roughly the same as for the LAMAN network, with extra computation costs required for the CHAMAN and FAMAN networks.

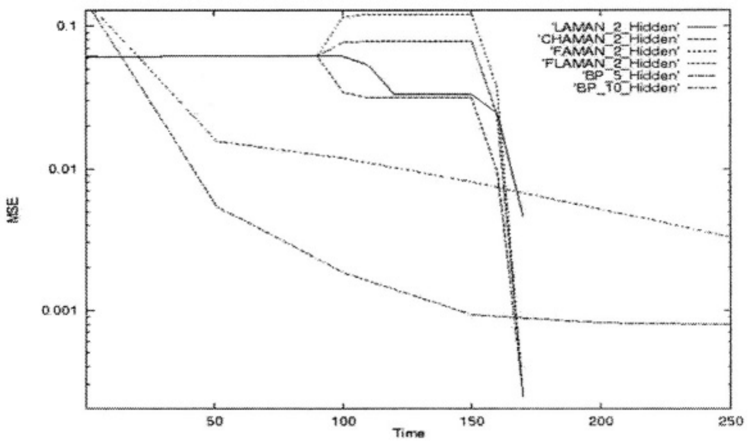


Fig. 2. Error graph of LAMAN, CHAMAN, FAMAN, and FLAMAN with 2 hidden nodes comparing with BP networks with 5 and 10 hidden nodes for Data Set 1

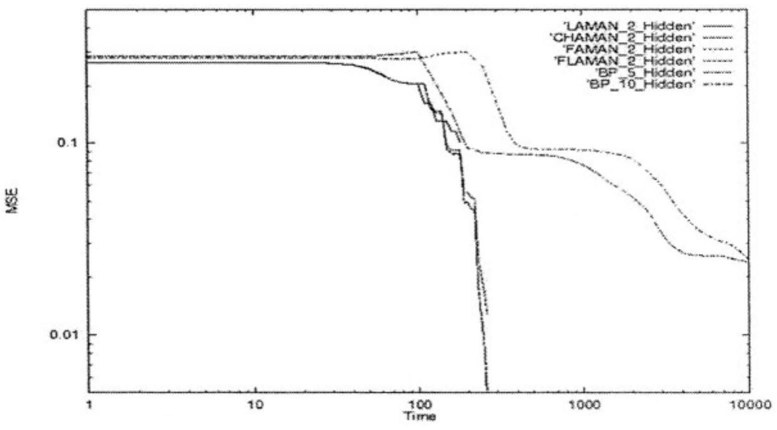


Fig. 3. Error graph of LAMAN, CHAMAN, FAMAN, and FLAMAN with 2 hidden nodes comparing with BP networks with 5 and 10 hidden nodes for Data Set 2

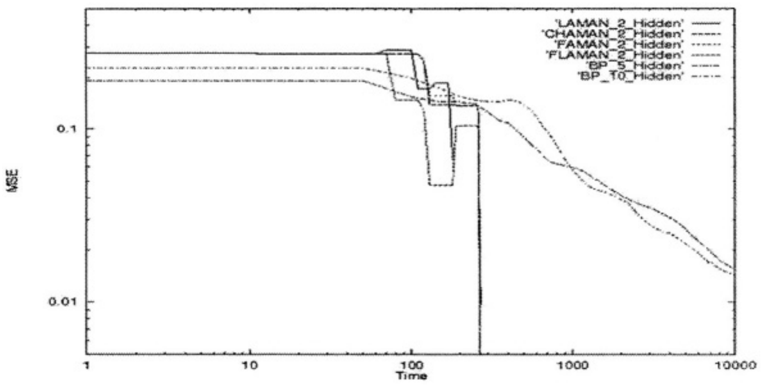


Fig. 4. Error graph of LAMAN, CHAMAN, FAMAN, and FLAMAN with 2 hidden nodes comparing with BP networks with 5 and 10 hidden nodes for Data Set 3

6 Conclusions

This paper proposed a novel approach to function approximation with several advantages over traditional monolithic networks. First, it does not require prior knowledge of the given problem. New modules are generated when the performance of current modules is inadequate. Second, it trains one module after another, so one module may be trained extensively for a difficult problem, whereas another module succeeds independently on an easier problem while requiring only a few training iterations. Third, the computation time requirements are very low since each module is trained on a small subset of the training data. Fourth, the proposed algorithms result in very low MSE and standard deviation since each module specializes on one region. When the data is inherently discrete and the mapping is region-specific, the performances of the proposed networks are significantly better than traditional approaches. The proposed approach can be applied to any regression problem characterized by some localized behavior, such as forecasting with financial data, where monolithic BP networks have produced poor results.

References

1. R. A. Jacobs, M. I. Jordan, and A. G. Barto. Task decomposition through competition in a modular connectionist architecture: The what and where vision task. *Cognitive Science,* 15:219.250, 1991.
2. S. E. Fahlman and C. Lebiere. The cascade-correlation learning architecture. *Advances in Neural Information Processing Systems II (Denver 1989),* pages 524.532, 1990.
3. Y. Le Cun and J.S. Denker and S.~A.~Solla. : Optimal Brain Damage, Advances in Neural Information Processing system 2, 598-605,1990
4. B.~Hassibi and D.~G.~Stork : econd Order Derivatives for Network Pruning: Optimal Brain Surgeon, Advances in Neural Information Processing Systems 5, 164-171,1993

5. G. Carpenter and S. Grossberg. The art of adaptive pattern recognition by a self-organizing neural network. *Computer,* March:77.88,1988.
6. M.~Frean.: The Upstart Algorithm: A Method for Constructing and Training Feedforward Neural Networks, Neural Computation 2,198-209,1990
7. J.-A.~Sirat and J.-P.~Nadal : Neural Trees: A New Tool for Classification, Preprint, Laboratoires d'Electronique Philips, Limeil-Brevannes, France", 1990
8. H. Lee and K. Mehrotra and C.K.Mohan and S. Ranka, An Incremental Network Construction Algorithm for Approximating Discontinuous Functions, Proc. Int'l. Conf. Neural Networks, IV, 2191-2196, 1994
9. D.C. Plaut and G.E. Hinton : Learning sets of filters using back-propagation. Computer Speech and Language, Vol.2, 35-61,1987
10. R. S. Sutton : Two problems with back-propagation and other steepest-descent learning procedures for networks. Proc. Eighth Annual Conference of the Cognitive Science Society. 823-832,1986
11. M. Sekiguchi and S. Nagata and K. Asakawa : Behavior Control for a Mobile Robot by Multi-Hierarchcal Neural Network, Proc. IEEE Int'l. Conf. Robotics and Automation, 1578-1583,1989
12. L. Y. Pratt and C. A. Kamm L : Improving Phoneme Classification Neural Network through Problem Decomposition, IEEE Int'l. Joint Conf. Neural, Vol. 2, 821-826, 1991
13. F. J. Smieja : Multiple network systems Minos modules : Task division and module discrimination, Proceedings of the 8th AISB conference on Artificial Intelligence, 1991
14. D. L. Reilly and C. Scofield and C. Elbaum and L. N. Cooper : Learning System Architectures Composed of Multiple Learning Modules, IEEE Int'l. Conf. Neural Networks, Vol. 2, 495-503, 1989
15. R.A. Jacobs and M.I. Jordan. Learning piecewise control strategies in a modular neural network architecture. *IEEE Trans. Systems, Man and Cybernetics,* 23:337.345,1993.
16. T. Kohonen. Improved versions of Learning Vector Quantization. In *Proc. Int'l. Joint Conf. Neural Networks, San Diego (CA),* volume I, pages 545.550, 1990.
17. R. Hecht-Nielsen : Counterpropagation networks. *IEEE International Conference on Neural Networks,* II:19.32, 1987b.
18. K. L.Clarkson : Safe and effective determinant evaluation. In *Proc. 31st IEEE Symposium on Foundations of Computer Science,* pages 387.395, 1992.
19. K. L.Clarkson, K.Mehlhorn, and R.Seidel. Four results on randomized incremental constructions. In *Comp. Geom.: Theory and Applications,* pages 185.121, 1993.
20. T. Nomura and T. Miyoshi. An adaptive rule extration with the fuzzy self-organizing map and a comparison with other methods. In *Proc. ISUMA-NAFIPS'95,* pages 311.316,1995.
21. T. Nomura and T. Miyoshi. An adaptive fuzzy rule extraction using hybrid model of the fuzzy selforganizing map and the genetic algorithm with numerical chromosomes. In *Proc. Fourth International Conference on Soft Computing (IIZUKA'96),* volume I, pages 70.73, 1996.

A Novel Hybrid Approach of Mean Field Annealing and Genetic Algorithm for Load Balancing Problem

Chuleui Hong[1], Wonil Kim[2], and Yeongjoon Kim[1]

[1] Software School, Sangmyung University, Seoul, Korea
{hongch, yjkim}@smu.ac.kr
[2] Dept. of Digital Contents, College of Electronics and Information Engineering, Sejong University, Seoul, Korea
wikim@sejong.ac.kr

Abstract. In this paper, we introduce a new solution for the load balancing problem, which is an important issue in parallel processing. Our novel load balancing technique called MGA is a hybrid algorithm of Mean Field Annealing (MFA) and Simulated annealing-like Genetic Algorithm (SGA). SGA uses the Metropolis criteria for state transition as in simulated annealing to keep the convergence property in MFA. The MGA combines the benefit of both the rapid convergence property of MFA and various and effective genetic operations of SGA. We compare the proposed MGA with MFA and GA. Our experimental results show that our new technique improves performance in terms of communication cost, load imbalance and maximum execution time.

1 Introduction

If tasks are not properly assigned to the processors in distributed memory multiprocessors, processor idle time and the interprocessor communication overhead from load imbalance may lead to poor utilization of parallel processors. This is a load balance mapping problem between tasks and processors [1,2,3,4,5]. Multiple tasks are allocated to the given processors in order to minimize the expected execution time of the parallel program

This paper presents the new solution of load balancing problem for distributed memory, message passing parallel computers. The proposed algorithm called MGA is a hybrid heuristic based on mean field annealing (MFA) [1, 4] and genetic algorithm (GA) [2, 5, 6]. MFA has the characteristics of rapid convergence to the equilibrium state while the simulated annealing (SA) [6, 7] takes long time to reach the equilibrium state. However, the general solution quality of MFA is worse than that of SA. Genetic algorithm has the powerful and diverse operations comparing with other heuristics. The proposed algorithm takes benefits of MFA and GA. Since genetic algorithm does not have a concept of temperature, genetic operations break the convergence property of mean field annealing. In the proposed method, the typical genetic algorithm is modified where the evolved new states are accepted by the Metropolis criteria as in simulated annealing. The modified Simulate annealing-like Genetic Algorithm is called SGA. SGA can keep the convergence property of MFA operations at each temperature. The simulation results show that the new MGA is better than MFA and GA. The proposed heuristic reduces interprocessor communication time, load imbalance among processors and expected maximum execution time.

Section 2 discusses the mapping problem in parallel processing. Section 3 proposes the novel hybrid approach called MGA. We present the formulation of MFA and SGA for load balancing problem. In Section 4, we explain the simulation environments of the proposed algorithm and evaluate the performance of MGA in comparison with MFA and GA. Finally, section 5 concludes.

2 The Mapping Problem in Multiprocessors

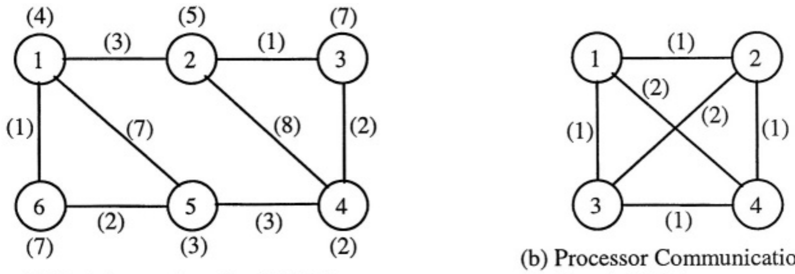
The multiprocessor mapping problem is a typical load balancing optimization problem. A mapping problem can be represented with two undirected graphs, called the Task Interaction Graph (TIG) and the Processor Communication Graph (PCG). TIG is denoted as $G_T(V, E)$. $|V| = N$ vertices are labeled as $(1, 2, ..., i, j, ..., N)$. Vertices of G_T represent the atomic tasks of the parallel program and its weight, w_i, denotes the computational cost of task i for $1 \le i \le N$. Edge E represents interaction between two tasks. Edge weight, e_{ij}, denotes the communication cost between tasks i and j that are connected by edge $(i, j) \in E$. The PCG is denoted as $G_P(P, D)$. G_P is a complete graph with $|P| = K$ nodes and $|D| = {}_KC_2$ edges. Vertices of the G_P are labeled as $(1, 2, ..., p, q, ..., K)$, representing the processors of the target multicomputers. Edge weight, d_{pq}, for $1 \le p,q \le K$ and $p \ne q$, denotes the unit communication cost between processor p and q.

The load balancing problem is to find a many-to-one mapping function M: $V \rightarrow P$. That is, each vertex of G_T is assigned to the unique node of G_P. Each processor is balanced in computational load *(Load)* while minimizing the total communication cost (*Comm*) between processors.

$$Comm = \sum_{(i,j)\in E, M(i)\ne M(j)} e_{ij}\, d_{M(i)M(j)}\,, \quad Load_p = \sum_{i\in V, M(i)=p} w_i\,, \quad 1 \le p \le K \tag{1}$$

$M(i)$ denotes the processor to which task i is mapped, i.e. $M(i)=p$ represents that task i is mapped to the processor p. In Eq (1), if tasks i and j in G_T are allocated to the different processors, i.e. $M(i) \ne M(j)$ in G_P, the communication cost occurs. The contribution of this to *Comm* is the multiplication of the interaction amount of task i and j, e_{ij}, and the unit communication cost of different processors p and q, d_{pq}, where $M(i)=p$ and $M(j)=q$. $Load_p$ denotes the summation of computational cost of tasks i, w_i, which are allocated processor p, $M(i)=p$.

Figure 1 shows an example of the mapping problem. Figure 1(a) represents TIG of N=6 tasks, and Figure 1(b) is for PCG of 2-dimensional mesh topology consisting of K=4 processors. The numbers in circles represent the identifiers of tasks and processor in Figure 1(a) and 1(b) respectively. In Figure 1(a), the weight of vertices and edges is for size of tasks and communications respectively. In Figure 1(b), the weight of edge represents the number of hops between two processors. Figure 1(c) shows the optimal task allocation to processors.



(a) Task Interaction Graph (TIG)

(b) Processor Communication Graph (PCG)

i	1	2	3	4	5	6
$M(i)$	1	2	4	2	1	3

(c) The Optimal Solution

Fig. 1. The Example of Mapping Problem

A spin matrix consists of N task rows and K processor columns. $N \times K$ spin matrix represents the allocation state. The value of spin element (i, p), s_{ip}, is the probability of mapping task i to processor p. So, the range of s_{ip} is $0 \le s_{ip} \le 1$ and the sum of each row is 1. The initial value of s_{ip} is $1/K$ and s_{ip} converges 0 or 1 as the solution state is reached eventually. $s_{ip} = 1$ means that task i is mapped to processor p. Figure 2 displays the initial and final optimal solution spin matrix of Figure 1.

	1	2	3	4
1	0.25	0.25	0.25	0.25
2	0.25	0.25	0.25	0.25
3	0.25	0.25	0.25	0.25
4	0.25	0.25	0.25	0.25
5	0.25	0.25	0.25	0.25
6	0.25	0.25	0.25	0.25

(a) The Initial State

	1	2	3	4
1	1	0	0	0
2	0	1	0	0
3	0	0	0	1
4	0	1	0	0
5	1	0	0	0
6	0	0	1	0

(b) The Solution State

Fig. 2. The Spin Matrix of Figure 1

The cost function, *C(s),* is set to minimize the total communication cost and to equally balance the computational load among processors of Eq (1).

$$C(s) = \sum_{i=1}^{N} \sum_{j \ne i} \sum_{p=1}^{K} \sum_{q \ne p} e_{ij} s_{ip} s_{jq} d_{pq} + r \sum_{i=1}^{N} \sum_{j \ne i} \sum_{p=1}^{K} s_{ip} s_{jp} w_i w_j \, . \qquad (2)$$

The first term of cost function, Eq (2), represents interprocessor communication cost (IPC) between two tasks i and j when task i and j are mapped to different processor p and q respectively. Therefore the first IPC term minimizes as two tasks with large interaction amount are mapped to the same processors. The second term of Eq (2) means the multiplication of computational cost of two tasks *i* and *j* mapped to the same processor *p*. The second computation term also minimizes when the computational costs of each processor are almost the same. It is the sum of squares of the amount of tasks in the same processor. The ratio *r* changes adaptively in the optimization process in order to balance the communication and computation cost. Changing the ratio *r* adaptively results in better optimal solution than fixing the ratio *r*. The optimal solution is to find the minimum of the cost function.

3 Load Balancing in Parallel Processing

3.1 Mean Field Annealing (MFA)

In this section, we discuss the concept of MFA for the mapping problem [1]. The optimization problem can be easily solved by the mean field approximation which estimates the state of a system of particles or spins in thermal equilibrium. The mean field annealing (MFA) is derived from simulated annealing (SA) based on mean field approximation method in physics. Though SA provides best solutions in various kinds of general optimization problems, it takes long time to obtain the solution. While SA changes the states randomly, MFA makes the system reach the equilibrium state very fast using the mean value estimated by mean field approximation.

```
<Procedure Mean Field Annealing>
Get the initial temperature $T_0$ , and set $T= T_0$
Initialize the spin average $s=[s_{11}, \ldots, s_{ip}, \ldots, s_{NK}]$
while temperature $T$ is in cooling range begin
    while cost change is less than $\varepsilon$ for
          continuous N annealing process begin
       Select a task $i$ at random
       Compute the mean fields of the spins at the $i$th row
          $\phi_{ip} = -\sum_{j \neq i}^{N} \sum_{q \neq p}^{K} e_{ij} s_{jq} d_{pq} - r \sum_{j \neq i}^{N} s_{jp} w_i w_j$   for $1 \leq p \leq K$
       Compute the new spin values at the $i$th row
          $s_{ip}^{new} = \frac{e^{\phi_{ip}/T}}{\sum_{p=1}^{K} e^{\phi_{ip}/T}}$
       Compute the cost change due to spin updates
          $\Delta C = \sum_{p=1}^{K} \phi_{ip} (s_{ip}^{new} - s_{ip})$
       Update the spin values at the $i$th row
          $s_{ip} = s_{ip}^{new}$        for $1 \leq p \leq K$
    end
    decrease the temperature
end
```

In implementing MFA, the cooling schedule has a great effect on the solution quality. Therefore the cooling schedule must be chosen carefully according to the characteristics of problem and cost function. The following parameters must be specified.

Initial Temperature (T_0): Initial temperature, T_0, is set such that the probability where the cost change is less than $\varepsilon(=0.5)$ is more than 95% for the number of tasks (N) annealing process

Final Temperature (T_f): T_f is set to the temperature where the value of the cost change is in $\varepsilon/1{,}000$ for continuous 20 temperature changes.

Length of the Markov Chain at a Certain Temperature T_k (L_k): L_k is the number of state transition to reach the equilibrium state. It is set to the number of state transitions where the cost change is less than ε for continuous N annealing process.

Decrement Strategy of the Temperature ($T_{k+1} = \alpha T_k$): A fixed decrement ratio, α, is set to 0.9 experimentally. This strategy decreases the temperature proportional to the logarithm of the temperature.

3.2 Simulated Annealing-Like Genetic Algorithm (SGA)

Genetic algorithm (GA) is a powerful heuristic search method based on an analogy to the biological evolution model. GA provides an effective means for global optimization in a complex search space such as NP-complete optimization including mapping problem, function optimization and etc.

Since GA does not have a concept of temperature comparing with MFA and SA, general GA breaks the convergence property of MFA in the proposed hybrid algorithm. We modified GA such that the new evolved state is accepted with a Metropolis criterion like simulated annealing in order to keep the convergence property of MFA. The modified GA is called SGA. Following is a pseudo code of SGA.

```
<Procedure SGA>
Initialize population from MFA spin matrix
Evaluate population
for number of tasks(=$N$) times begin
    Select individuals from current population
    Reproduce next population
    for select 2 individuals by turns begin
        Perform crossover with probability of crossover
        Calculate the cost change ($\Delta C$)
        if exp(-$\Delta C/T$) > random[0,1] then
            Accept new individuals
    end
    for all individuals begin
        Perform mutation with probability of mutation
        Calculate the cost change ($\Delta C$)
        if exp(-$\Delta C/T$) > random[0,1] then
            Accept new individuals
    end
    Keep the best individual
end
```

The followings are parameters for SGA implementation.

Representation of Individual: The individual is represented by a string in the order of tasks whose value is allocated processor identification. For example, a string, "1,3,4,1,2", means that tasks are allocated to processors such that task 1 to processor 1, task 2 to processor 3, task 3 to processor 4, task 4 to processor 1, task 5 to processor 2.

Form GA's Population from MFA's Spin Matrix: The individuals are generated randomly with the probability as same as that of spin matrix in MFA. For example, if spin values of an arbitrary i^{th} task, which is the elements of i^{th} row, is 0.2, 0.4, 0.1, 0.1, 0.2, an individual is made such that the i^{th} character in a string can be 1 with a probability of 0.2, 2 with that of 0.4, 3 with that of 0.1, 4 with that of 0.1 and so on.

The Size of Population : In the experiment, the number of individuals in a population is set to 100.

The Cost or Objective Function : The linear cost function is chosen as same as that of MFA.

Selection : A proportionate selection scheme is used. The individual is chosen randomly with a ratio of individual's cost over sum of population's cost.

Reproduction : The new population is made with individuals chosen by selection operation.

Crossover : After 2 individuals are chosen by turns from population, the part of string is exchanged. So the tasks allocations to processors are exchanged. The size of exchanged part is randomly set less than 1/4 of string length. The probability of crossover is 0.8.

Mutation : An individual is selected with a probability of 0.05. Exchange and displacement operation is implemented on the selected individual. The probability of exchange is 0.1 and displacement is 0.9. The exchange operation selects less than 4 tasks randomly in an individual and then allocations to processors of selected tasks are exchanged. The displacement operation selects 1 or 2 tasks randomly and changes their allocations to processors randomly.

Keeping Best Individual : The individual with the lowest cost is kept in the next population. This prevents genetic operations from losing the best solution of the previous stage.

Termination Condition : The number of genetic operation in a certain temperature is set to the number of tasks. This number is set experimentally.

Simulated Annealing-Like Genetic Algorithm (SGA) : In order to keep the thermal equilibrium of MFA, the new configurations generated by genetic operations are accepted or rejected by the Metropolis Criteria which is used in SA.

$$\Pr[\Delta C \text{ is accepted}] = \min\left(1, \exp\left(\frac{\Delta C}{T}\right)\right). \tag{3}$$

ΔC is the cost change of new state from old state which is made by subtracting the cost of new state from that of old one. T is the current temperature.

Form the MFA's Spin Matrix from GA's Population : This is inverse operation of making GA's population from MFA's spin matrix. After GA finishes, selection and reproduction are applied on the population for the new generation, and then a spin matrix is made on the ratio of task allocations to processors. For example, let the population size be 10 and the problem be as same as Fig.1 where there are 6 tasks and 4 processors. The i^{th} character in an individual represents the allocation of task i. If the i^{th} characters in 10 individuals are 1,2,3,4,1,2,3,1,2,1 respectively, the probabilities of task i to processor 1, 2, 3, and 4 are 0.4, 0.3, 0.2, 0.1 respectively also. So, the i^{th} row of a spin matrix is set to 0.4, 0.3, 0.2, 0.1 according to the distribution of task i in GA population.

3.3 MGA Hybrid Algorithm

A new hybrid algorithm called MGA combines the merits of mean field annealing (MFA) and simulated annealing-like genetic algorithm (SGA). MFA can reach the thermal equilibrium faster than simulated annealing and GA has powerful and various genetic operations such as selection, crossover and mutation.

First, MFA is applied on a spin matrix to reach the thermal equilibrium fast. After the thermal equilibrium is reached, the population for GA is made according to the distribution of task allocation in the spin matrix. Next, GA operations are applied on the population while keeping the thermal equilibrium by transiting the new state with Metropolis criteria. MFA and GA are applied by turns until the system freeze.

```
<MGA Hybrid Algorithm>
Initialize mapping problems /* getting TIG and PCG */
Forms the spin matrix, $s = [s_{11}, \ldots, s_{ip}, \ldots, s_{NK}]$
Set the initial ratio $r$
Get the initial temperature $T_0$, and set $T = T_0$
while $T \geq T_f$ begin
    Executes MFA
    Forms GA population from a spin matrix of MFA
    Executes SGA
    Forms the spin matrix of MFA from GA population
    Adjusts the ratio $r$
    $T = \bullet \times T$
end
```

4 Simulation Results

The proposed MGA hybrid algorithm is compared with MFA and GA that use the same cost function as that of MGA. In this simulation, the size of tasks is 200 and 400. The multiprocessors are connected with wrap-around mesh topology. The computational costs of each task are distributed uniformly ranging [1..10]. The communication costs between any two tasks ranges [1..5] with uniform distribution. The number of communications is set to 1, 2, or 3 times of the number of tasks. Each experiment is implemented 20 times varying the seed of random number generator and TIG representing the computational and communication cost.

The coefficient r in a linear cost function is for balancing the computation and communication cost between processors. The initially task allocations to processors are uniformly distributed and accordingly a spin matrix is generated. The initial ratio r is computed as in Eq (4). As the temperature decreases, the coefficient r changes adaptively according to Eq (5) in order to reflect the changed interprocessor communication cost. r_{old} is the ratio used at the previous temperature and r_{new} is the newly calculated ratio at the current temperature.

$$r_{init} = \frac{\text{Comm. cost}}{\#\text{ of processors} \times \text{Comp. cost}} = \frac{\sum_{i=1}^{N}\sum_{j\neq i}\sum_{p=1}^{K}\sum_{q\neq p} e_{ij}\, s_{ip}\, s_{jq}\, d_{pq}}{K \times \sum_{i=1}^{N}\sum_{j\neq i}\sum_{p=1}^{K} s_{ip}\, s_{jp}\, w_i\, w_j} \qquad (4)$$

$$r_{new} = \begin{cases} 0.9 \times r_{old} & \text{if } r_{new} < r_{old} \\ r_{old} & \text{Otherwise} \end{cases} \qquad (5)$$

Table 1. The maximum completion times when the initial value of r is kept and r changes adaptively according to Eq (5)

Problem Size			MFA		MGA	
N	|E|	K	Keep *r*	Change *r*	Keep *r*	Change *r*
200	200	16	336.65	138.15	134.65	120.2
	400	16	707.45	525.85	324.15	320.1
	600	16	845	767.15	526.75	544.25
	200	36	193.35	84.65	90.3	71.95
	400	36	430.5	307.25	235.8	218.45
	600	36	651.55	559.15	420	412.15
400	400	16	627.1	279.05	256.4	222.75
	800	16	1189.95	888.1	617.15	587
	1200	16	1862.8	1557.4	971.9	987.5
	400	36	410.5	152.85	143.85	128.65
	800	36	834.45	617	410.9	385.15
	1200	36	1376.8	1065.5	714.65	692.95

Table 1 compares the maximum completion times when the initial value of r is kept and r changes adaptively according to Eq (5). The completion time of an arbitrary processor is the sum of computation cost of its processor and communication cost with other processors, which represents time for a processor to complete its all allocated tasks. The maximum completion time is defined as the maximum value among the completion times of all processors. In Table 1, N is the number of tasks, $|E|$ is the total number of interprocessor communications, and K represents the number of processors. The averaged performance improvement in MFA is better than that in MGA. It is guessed that genetic operations of MGA have a great contribution to find better solutions. So, changing r has a less effect on the solution quality of MGA than that of MFA.

Table 2. Total interprocessor communication cost and percent computational cost imbalance

Problem Size			Total Comm. Time			Comp. Cost Imbalance		
N	|E|	K	MFA	MFA	MGA	MFA	MFA	MGA
200	200	16	414.6	627.55	346.8	47%	47%	39%
	400	16	3650.5	3260.9	2084	68%	68%	61%
	600	16	6525.6	5744	4215.8	65%	65%	60%
	200	36	539	1718.1	490.5	66%	66%	58%
	400	36	4002.2	4903.4	2956.3	85%	85%	77%
	600	36	8608.2	8546.1	6231.2	85%	85%	78%
400	400	16	994.3	2370.5	629.7	47%	47%	32%
	800	16	8089.1	6714.6	4004.2	61%	61%	57%
	1200	16	14677	11743	8348.4	49%	49%	54%
	400	36	1062.7	3539.5	852.4	60%	60%	50%
	800	36	10021	10360	5603	79%	79%	72%
	1200	36	19937	17780	11868	75%	75%	70%
		Avg.	6543.4	6442.3	3862.8	66%	66%	59%

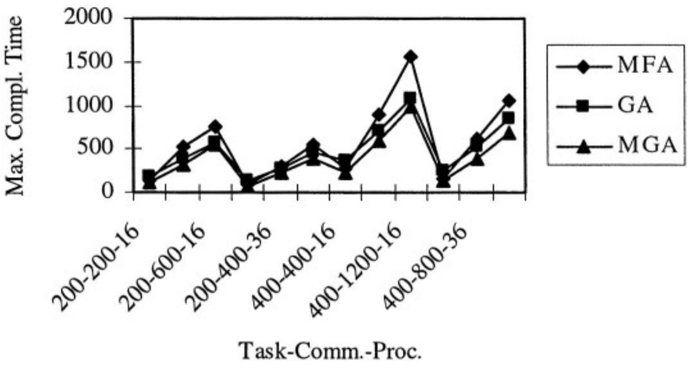


Fig. 3. Maximum completion time for various problem sizes

Table 2 displays averaged total interprocessor communication cost of each algorithm. The average performance improvement from MFA to MGA, which is the percent reduction of communication cost normalized by that of MFA, is 33%. It is 45% from GA to MGA. Table 2 also displays computational cost imbalance which is defined as a difference between maximum and minimum computational cost of processors normalized by the maximum cost. The computational cost imbalance of each algorithm displays a little difference, while total communication costs are much more different. This implies that the interprocessor communication cost has a greater effect on the solution quality than the computational cost.

Finally the averaged execution time of MGA is 1.5 and 1.7 times longer than that of MFA and GA respectively. This is a trade-off between the solution quality and execution time. Figure 3 displays the maximum completion time of MFA, GA and MGA for various problems. This shows that MGA is more useful as the problem size increases.

5 Conclusions

In this paper, we proposed a new hybrid heuristic called MGA. The proposed approach combines the merits of MFA and GA on a load balancing problem in parallel distributed memory multiprocessor systems. This new hybrid algorithm is compared and evaluated with MFA and GA. The solution quality of MGA is superior to that of MFA and GA while execution time of MGA takes longer than the compared methods. There can be the trade off between the solution quality and execution time by modifying the cooling schedule and genetic operations. MGA was also verified more promising and useful as the problem size increases and the problem is more complex.

The proposed algorithm can be easily developed as a parallel algorithm since MFA and GA can be parallelized easily. This algorithm also can be applied efficiently to the broad range of NP-Complete problems.

References

1. Susmita, S., Bhargab, S., Bhattacharya, B.: Manhattan-diagonal routing in channels and switchboxes. ACM Trans on DAES, Vol. 09, No. 01. (2004)
2. Bultan, T., Aykanat, C. : A New Mapping Heuristic Based on Mean Field Annealing. Journal of Parallel & Distributed Computing,16 (1992) 292-305
3. Heiss, H.-U., Dormanns, M. : Mapping Tasks to Processors with the Aid of Kohonen Network. Proc. High Performance Computing Conference, Singapore (1994) 133-143
4. Park, K., Hong, C.E. : Performance of Heuristic Task Allocation Algorithms. Journal of Natural Science, CUK, Vol. 18 (1998) 145-155
5. Salleh, S., Zomaya, A. Y.: Multiprocessor Scheduling Using Mean-Field Annealing. Proc. of the First Workshop on Biologically Inspired Solutions to Parallel Processing Problems (BioSP3) (1998) 288-296
6. Zomaya, A.Y., Teh, Y.W.: Observations on Using Genetic Algorithms for Dynamic Load-Balancing. IEEE Transactions on Parallel and Distributed Systems, Vol. 12, No. 9 (2001) 899-911

7. Hong, C.E.: Channel Routing using Asynchronous Distributed Genetic Algorithm. Journal of Computer Software & Media Tech., SMU, Vol. 2. (2003)
8. Hong, C., McMillin, B.: Relaxing synchronization in distributed simulated annealing. IEEE Trans. on Parallel and Distributed Systems, Vol. 16, No. 2. (1995) 189-195

Geodesic Topographic Product: An Improvement to Measure Topology Preservation of Self-Organizing Neural Networks

Francisco Flórez Revuelta, Juan Manuel García Chamizo, José García Rodríguez, and Antonio Hernández Sáez

Departamento de Tecnología Informática y Computatión, Universidad de Alicante, Apdo. 99. 03080 Alicante, España
{florez, juanma, jgarcia, ahernandez}@dtic.ua.es
http://www.ua.es/i2rc

Abstract. Self-organizing neural networks endeavour to preserve the topology of an input space by means of competitive learning. There are diverse measures that allow to quantify how good is this topology preservation. However, most of them are not applicable to measure non-linear input manifolds, since they don't consider the topology of the input space in their calculation. In this work, we have modified one of the most employed measures, the topographic product, incorporating the geodesic distance as distance measure among the reference vectors of the neurons. Thus, it is possible to use it with non-lineal input spaces. This improvement allows to extend the studies realized with the original topographic product focused to the representation of objects by means of self-organizing neural networks. It would be also useful to determine the right dimensionality that a network must have to adapt correctly to an input manifold.

1 Introduction

A self-organizing neural network $\mathcal{A}$ consists of a set of $\mathcal{N}$ neurons, each one of them has associated a reference vector belonging to the input space $w_i \in \mathbf{R}^n$. These neurons are connected by edges to determine the topology of the network. By means of a competitive process, it is carried out an adaptation of the reference vectors as well as of the interconnection network among them; obtaining a mapping ψ that tries to preserve the topology of an input manifold $\mathcal{M} \subseteq \mathbf{R}^n$, associating to each one of the input patterns $\xi \in \mathcal{M}$ its closest neuron $i_\xi \in \mathcal{A}$:

$$\left\| w_{i_\xi} - \xi \right\| \leq \left\| w_j - \xi \right\|, \quad \forall j \in \mathcal{A} \tag{1}$$

Researchers have usually considered this mapping as topology-preserving. However, Martinetz and Schulten [1] limited this quality, by defining a Topology Representing Network as the neural network whose mapping $\psi_{\mathcal{M}\to\mathcal{A}}$ and inverse mapping $\psi_{\mathcal{A}\to\mathcal{M}}$, defined as:

$$\begin{aligned} \psi_{\mathcal{M}\to\mathcal{A}} &: \mathcal{M}\to\mathcal{A}; \quad \xi\in\mathcal{M}\hbar \quad i_{\xi}\in\mathcal{A} \\ \psi_{\mathcal{A}\to\mathcal{M}} &: \mathcal{A}\to\mathcal{M}; \quad i\in\mathcal{A}\hbar \quad w_i\in\mathcal{M} \end{aligned} \tag{2}$$

are topology-preserving, so that input patterns that are close within the input manifold $\mathcal{M}$ are mapped to neurons that are close within $\mathcal{A}$; and neurons close in $\mathcal{A}$ have associated similar reference vectors in $\mathcal{M}$.

There are many self-organizing models that only preserve the topology of an input space when both dimensionalities agree. So, if we don't know a priori the dimensionality of the input space, the learning of networks with different structures is usually carried out, chosen the one having the bigger degree of topology preservation.

In order to quantify this capacity of topology preservation, there have been developed diverse measures [2] [13] that are based on the evaluation of the positions of the neurons inside the network, as well as in the positions of the associated reference vectors. Among them, the most used is the topographic product [3]. However, except for the topographic function [4], no measure takes into consideration the form of the input manifold, giving similar results for mappings that clearly possess different qualities.

Since we want to apply the result of the neural network adaptation to the representation of objects [7] and their motion [14], it is important that the result preserves their topology correctly. For this reason, we need that the measure that quantifies it were adapted to this goal. So, we present a modification of the topographic product to be used to every type of subspace, linear and non-linear.

2 Geodesic Topographic Product

The topographic product was one of the first attempts of quantifying the topology preservation of self-organizing neural networks. This measure is used to detect deviations between the dimensionalities of the network and that of the input space, detecting folds in the network and, indicating that is trying to approximate to an input manifold with different dimension.

2.1 Topographic Product

This measure compares the neighbourhood relationship between each pair of neurons in the network with respect to both their position on the map ($\mathcal{P}_2(j,k)$) and their reference vectors ($\mathcal{P}_1(j,k)$):

$$\mathcal{P}_1(j,k)=\left(\prod_{l=1}^{k}\frac{d^{V}\left(w_j,w_{n_l^{\mathcal{A}}(j)}\right)}{d^{V}\left(w_j,w_{n_l^{V}(j)}\right)}\right)^{1/k} \tag{3}$$

$$\mathcal{P}_2(j,k)=\left(\prod_{l=1}^{k}\frac{d^{\mathcal{A}}\left(j,n_l^{\mathcal{A}}(j)\right)}{d^{\mathcal{A}}\left(j,n_l^{V}(j)\right)}\right)^{1/k} \tag{4}$$

where j is a neuron, w_j is its reference vector, n_l^V is the l -th closest neighbour to j in the input manifold $\mathcal{V}$ according to a distance $d^{\mathcal{V}}$ and n_l^A is the l -th nearest neuron to j in the network $\mathcal{A}$ according to a distance $d^{\mathcal{A}}$.

Combining (3) and (4) a measure of the topological relationship between the neuron j and its k closer neurons is obtained:

$$P_3(j,k) = \left(\prod_{l=1}^{k} \frac{d^{\mathcal{V}}\left(w_j, w_{n_l^{\mathcal{A}}(j)}\right)}{d^{\mathcal{V}}\left(w_j, w_{n_l^{\mathcal{V}}(j)}\right)} \cdot \frac{d^{\mathcal{A}}\left(j, n_l^{\mathcal{A}}(j)\right)}{d^{\mathcal{A}}\left(j, n_l^{\mathcal{V}}(j)\right)} \right)^{1/2k} \tag{5}$$

To extend this measure to all the neurons of the network and all the possible neigbourhood orders, the topographic product P is defined as

$$P = \frac{1}{N(N-1)} \sum_{j=1}^{N} \sum_{k=1}^{N-1} \log\left(P_3(j,k)\right) \tag{6}$$

The topographic product takes different values if the dimension of the network is bigger ($P > 0$), similar ($P \approx 0$) or smaller ($P < 0$) than the dimension of the input manifold to which it has been adapted.

2.2 Generalization of the Topographic Product

The topographic product is usually employed taking as distance $d^{\mathcal{V}}$ the Euclidean distance among the reference vectors of the neurons, without considering the domain of the input manifold, and as distance $d^{\mathcal{A}}$ the length of the shortest path between two neurons inside the graph that constitutes the network. In [2] it is indicated that the result would be improved if the domain of the input manifold was contemplated in the calculation of the distance $d^{\mathcal{V}}$. From this idea, we take as $d^{\mathcal{V}}$ the geodesic distance (GD) [5], defined as the length of the minimum path that connects the reference vectors of two neurons inside the input manifold (figure 1a). If a path cannot be settled down between them, $d^{\mathcal{V}} = \infty$.

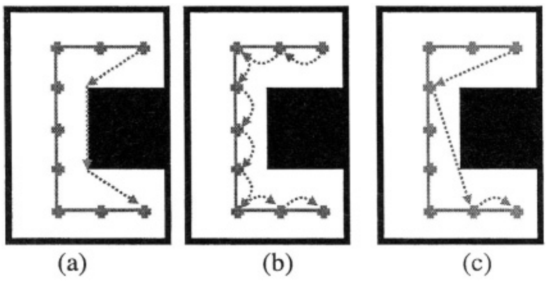


Fig. 1. Calculation of the geodesic distance

However, the calculation cost of the geodesic distance is very high. For it, two approaches to simplify their calculation have been carried out, using the obtained neural network:

- GD2: Calculation of the shortest path between each pair of neurons, contemplating the length of the edges that communicate them. This approach can be obtained, for example, with the Dijkstra algorithm (figure 1b). The result of this heuristic is similar to the one obtained with an approach presented in [15].
- GD3: Extension of the original graph by connecting every pair of neurons whose actual or new connection is completely inside the input manifold, and applying later the same process of the previous approach to the obtained network (figure 1c).

3 Comparison Between the Topographic Product and the Geodesic Topographic Product

An example that clearly illustrates the problem of measuring the topology preservation of a non linear manifold is shown in figure 2. Since the topographic product does not consider the topology of the manifold, the result is identical in both cases ($\mathcal{P} = -29.92 \times 10^{-3}$), being unable to identify which of both adaptations is better. Using the geodesic distance in the calculation of the topographic product, in case (a) a value close to 0 is obtained ($\mathcal{P} = 0.46 \times 10^{-3}$), indicating a correct topology preservation. In example (b), it is obtained the same value that only using the topographic product ($\mathcal{P} = -29.92 \times 10^{-3}$), indicating a bad adaptation to the input manifold. This fact is reflected in figure 3, where it is shown how the partial functions of the calculation of the topographic product ($\mathcal{P}_1$, $\mathcal{P}_2$ y $\mathcal{P}_3$) deviate from 1, indicating a nonexistent loss in topology preservation. This does not happen when using the geodesic distance in their calculation.

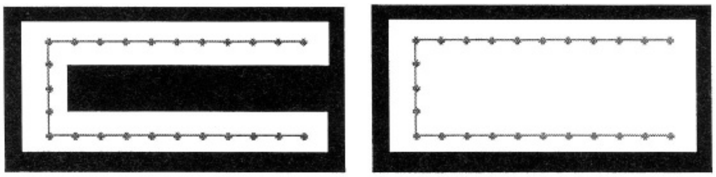


Fig. 2. Adaptation to (a) non-linear and (b) linear input manifolds

We have also applied the Geodesic Topographic Product to test the adaptation of different self-organizing neural models (Self-Organizing Maps (SOM) [8], Growing Cell Structures (GCS) [9], Neural Gas (NG) [10] and Growing Neural Gas (GNG) [11]) to diverse input manifolds (figure 4). Using the improved measure we have obtained appropriate results to the adaptations presented, indicating worse adaptations of the Kohonen self-organizing maps due to their interconnection structure and a more restricted learning process. This fact cannot be measured with the original Topographic Product (table 1, shaded cells), given that it just works accurately when used with linear spaces. There are also shown the results (GP2 and GP3) obtained using the two heuristics contemplated for the calculation of the geodesic distance. As it is observed, the measure of topology preservation does not vary substantially.

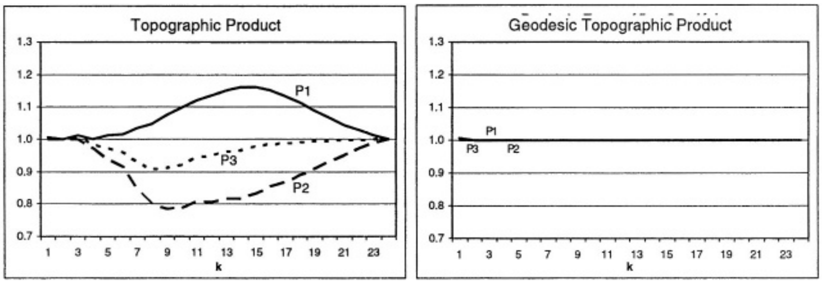


Fig. 3. Components $\mathcal{P}_1(k)$, $\mathcal{P}_2(k)$ and $\mathcal{P}_3(k)$ of the Topographic Product and the Geodesic Topographic Product

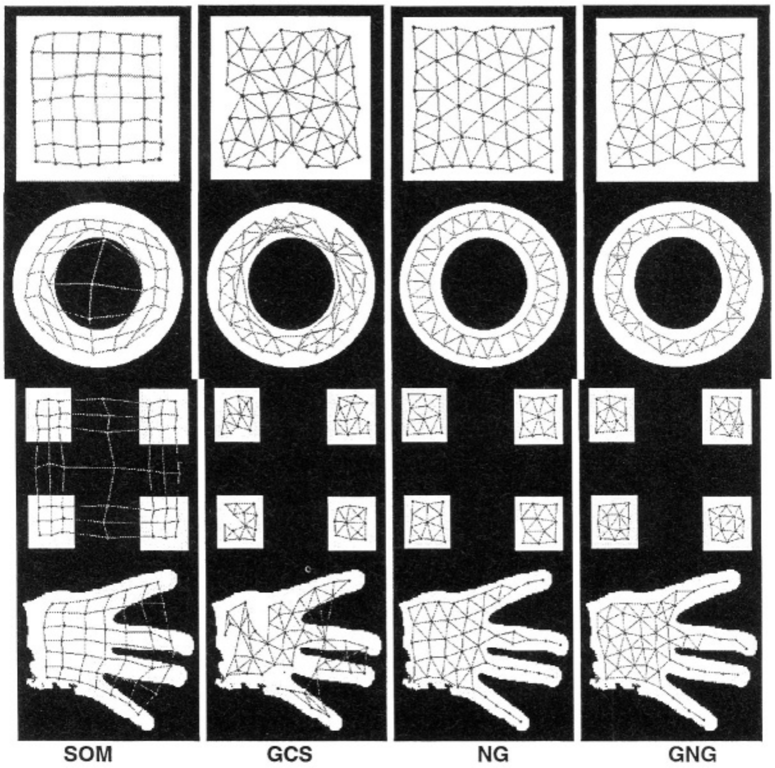


Fig. 4. Adaptations of Different Self-Organizing Neural Networks to Bidimensional Spaces

Table 1. Topology Preservation of the Different Self-Organizing Models

	Model	P	GP[1]	GP2	GP3
Square	SOM	5.91×10^{-3}	5.99×10^{-3}	6.75×10^{-3}	5.91×10^{-3}
	GCS	15.63×10^{-3}	15.71×10^{-3}	14.05×10^{-3}	15.63×10^{-3}
	NG	5.18×10^{-3}	5.22×10^{-3}	4.24×10^{-3}	5.18×10^{-3}
	GNG	7.79×10^{-3}	7.81×10^{-3}	7.04×10^{-3}	7.79×10^{-3}
Ring	SOM	7.99×10^{-3}	47.39×10^{-3}	39.26×10^{-3}	41.09×10^{-3}
	GCS	17.10×10^{-3}	19.92×10^{-3}	16.92×10^{-3}	18.63×10^{-3}
	NG	3.23×10^{-3}	3.69×10^{-3}	3.40×10^{-3}	3.70×10^{-3}
	GNG	4.57×10^{-3}	4.90×10^{-3}	4.61×10^{-3}	4.93×10^{-3}
4 squares	SOM	15.54×10^{-3}	-127.51×10^{-3}	-95.49×10^{-3}	-119.24×10^{-3}
	GCS	38.36×10^{-3}	6.44×10^{-3}	6.42×10^{-3}	6.02×10^{-3}
	NG	35.25×10^{-3}	2.91×10^{-3}	2.89×10^{-3}	2.46×10^{-3}
	GNG	35.61×10^{-3}	4.23×10^{-3}	4.22×10^{-3}	3.93×10^{-3}
Hand	SOM	5.99×10^{-3}	-2.15×10^{-3}	-17.42×10^{-3}	-2.84×10^{-3}
	GCS	12.83×10^{-3}	22.29×10^{-3}	8.57×10^{-3}	20.80×10^{-3}
	NG	8.06×10^{-3}	4.12×10^{-3}	4.14×10^{-3}	3.35×10^{-3}
	GNG	8.76×10^{-3}	6.10×10^{-3}	6.03×10^{-3}	5.65×10^{-3}

5 Comparison Between the Geodesic Topographic Product and the Topographic Function

As we have previously expressed, the original topographic product establishes the relations among the neurons according to the Euclidean distance, without taking into consideration the structure of the input space. So, it can not be applied to non-linear spaces. On the other hand, the topographic function [4] evaluates the degree of topology preservation depending on the similarity between the network and its induced Delaunay triangulation[2]. This allows its employment with all kinds of input spaces.

5.1 Topographic Function

Topology preservation of mappings $\psi_{\mathcal{M}\to\mathcal{A}}$ and $\psi_{\mathcal{A}\to\mathcal{M}}$ is respectively expressed as

$$f_j(k) = \#\left\{ j / d_{T_{\mathcal{A}}(i)}(i,j) > k ; d_{T_{\mathcal{MA}}(i)}(i,j) = 1 \right\} \tag{7}$$

[1] To make the calculation possible, we take a high value for $d^{\mathcal{V}}$ instead of $d^{\mathcal{V}} = \infty$ in case that there is not a path between the neurons or a neuron is placed out of the input manifold.

[2] Induced Delaunay triangulation is a subgraph of the Delaunay triangulation whose edges are completely into the input manifold $\mathcal{M}$.

$$f_j(-k) = \#\left\{ j / d_{T^{*}_{\mathcal{A}}(i)}(i,j) = 1; d_{T_{\mathcal{M}}^{\mathcal{A}}(i)}(i,j) > k \right\} \tag{8}$$

where $\#\{\bullet\}$ represents the cardinality of the set, $k = 1,\ldots,\mathcal{N}-1$ and $d_{T(i)}(i,j)$ is the distance between a pair of neurons based on different topological spaces[3] created from $\mathcal{M}$ and $\mathcal{A}$.

Topographic function $\Phi^{\mathcal{M}}_{\mathcal{A}}$ of mapping ψ is defined as:

$$\Phi^{\mathcal{M}}_{\mathcal{A}}(k) = \begin{cases} \frac{1}{\mathcal{N}} \sum_{j \in \mathcal{A}} f_j(k) & k \neq 0 \\ \Phi^{\mathcal{M}}_{\mathcal{A}}(1) + \Phi^{\mathcal{M}}_{\mathcal{A}}(-1) & k = 0 \end{cases} \tag{9}$$

The shape of $\Phi^{\mathcal{M}}_{\mathcal{A}}$ expresses detailed information about the magnitude of the distortions that appear in the map in relation to the input manifold $\mathcal{M}$. If so much information is not necessary, the sign of $\overline{\Phi} = \Phi^{\mathcal{M}}_{\mathcal{A}}(1) - \Phi^{\mathcal{M}}_{\mathcal{A}}(-1)$ indicates if the dimension of $\mathcal{A}$ is greater or lower than the one of $\mathcal{M}$.

5.2 Geodesic Topographic Product Vs. Topographic Function

Topographic function is able of measuring topology preservation of adaptations to non-linear spaces, since it takes into account the shape of the input space. Figure 5 shows the results of the application of the topographic function $\Phi^{\mathcal{M}}_{\mathcal{A}}$ to the adaptations presented in figure 2. Right graph shows that there is a loss in topology preservation for positive values of k, indicating that the network has less dimensionality than the input space. This fact does not occur when been applied to the left figure, where the network is correctly adapted. However, it doesn't obtain a single value as the topographic product does, but a function for the range $[-(\mathcal{N}-1),(\mathcal{N}-1)]$ is obtained. This makes comparisons between measures to different adaptations more complicated. The respective values of $\overline{\Phi}$ for the non-linear and the linear spaces presented in figure 2 are 0.2 and 1.2. These positive values indicate that the dimensionality of the networks is lower than the one of the input manifold, losing the information given by $\Phi^{\mathcal{M}}_{\mathcal{A}}$ about the magnitude of the distortions.

We have also applied both measures to study how topology preservation varies throughout the adaptive process of a self-organizing neural network [16]. Figure 6 shows topology preservation of the adaptation of a GNG to the "hand" bidimensional manifold presented in figure 4. Simplicity of the calculation of the topographic function causes its value to be more sensible to little variations of the network, producing big fluctuations in its results, which does not happen with the geodesic topographic product.

[3] In [4] and [12] there are explanations about how these spaces are created depending on the topology, lattice or not predefined, of the neural network.

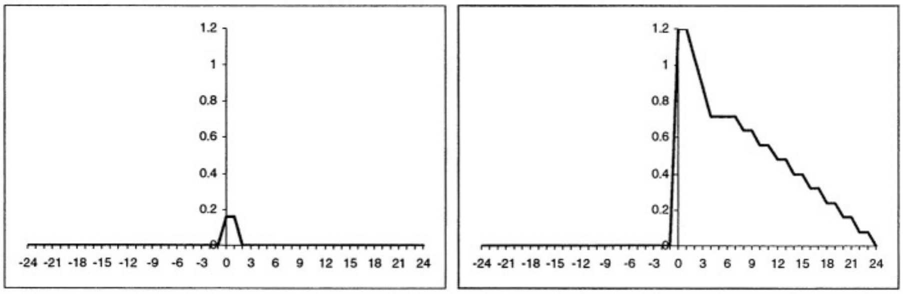


Fig. 5. Topology preservation measured with the topographic function for the adaptations presented in figure 2

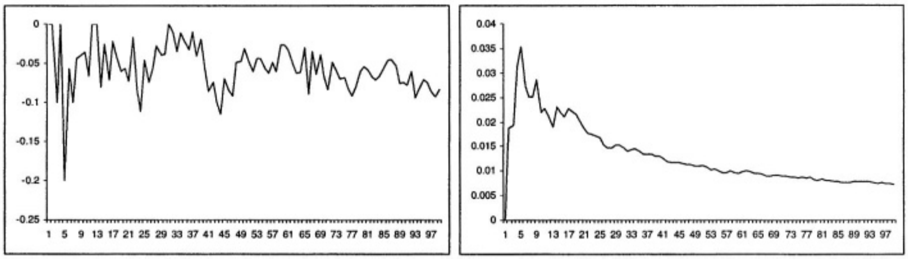


Fig. 6. Topology preservation during the adaptive growing process of the GNG measured with the topographic function (left) and the geodesic topographic product (right). Horizontal axis shows the size (number of neurons) of the network

6 Conclusions and Future Works

The use of the geodesic distance as a measure of distance between each pair of neurons into the calculation of the topographic product, allows its use to measure topology preservation of the adaptation of self-organizing neural networks to non-linear input spaces. This makes possible that other works, in which there has been used the original topographic product could be extended to non-linear manifolds. Given the high cost of its calculation, diverse heuristics have been presented, obtaining similar results.

The interest of developing a measure adapted to all kinds of spaces departs from the work that is been realized directed to the design and development of self-organizing neural networks for the representation of objects [7] and their motions [14]. By this reason, this new measure will allow the correct calibration of the topology preservation of different representations.

Nowadays, we are working with different self-organizing models (Neural Gas, Growing Neural Gas, GWR [12]), studying their degree of topology preservation throughout the adaptation process depending on the learning parameters. From the obtained results, we want to extract which are the characteristics of these networks that allow a suitable and fast representation of an input space, for their employment in these applications and, also, for the development of new self-organizing models based on the combination of them.

Acknowledgements

This work has been partially funded by the Science and Technology Agency of the Valencian Autonomous Government (Spain) under project CTIDIA/2002/109.

References

1 Martinetz, T., Schulten, K.: Topology Representing Networks. Neural Networks, 7(3) (1994) 507-522
2. Bauer, H.-U., Herrmann, M., Villmann, T.: Neural Maps and Topographic Vector Quantization. Neural Networks, 12(4-5) (1999) 659-676
3. Bauer, H.-U., Pawelzik, K.R.: Quantifying the Neighborhood Preservation of Self-Organizing Feature Maps. IEEE Transactions on Neural Networks, 3(4) (1992) 570-578
4. Villmann, T., Der, R., Herrmann, M., Martinetz, T.M.: Topology Preservation in Self-Organizing Feature Maps: Exact Definition and Measurement. IEEE Transactions on Neural Networks, 8(2) (1997) 256-266
5. Sonka, M., Hlavac, V., Boyle, R.: Image Processing, Analysis, and Machine Vision. 2nd edition. Brooks/Cole Publishing Company (1998)
6. Flórez, F., García, J.M., García, J., Hernández, A.: Representing 2D-objects. Comparison of several self-organizing networks. In Advances in Neural Networks World. WSEAS Press (2002) 69-72
7. Flórez, F., García, J.M., García, J., Hernández, A.: Representation of 2D Objects with a Topology Preserving Network. In Proceedings of the 2nd International Workshop on Pattern Recognition in Information Systems (PRIS'02), Alicante. ICEIS Press (2001) 267-276
8. Kohonen, T.: Self-Organizing Maps. Springer-Verlag, Berlin Heidelberg (1995)
9. Fritzke, B.: Growing Cell Structures – A Self-organizing Network for Unsupervised and Supervised Learning. Technical Report TR-93-026, International Computer Science Institute, Berkeley, California (1993)
10. Martinetz, T, Schulten, K.: A "Neural-Gas" Network Learns Topologies. In Artificial Neural Networks, T. Kohonen, K. Mäkisara, O. Simula y J. Kangas (eds.) (1991) 1:397-402
11. Fritzke, B.: A Growing Neural Gas Network Learns Topologies. In Advances in Neural Information Processing Systems 7, G. Tesauro, D.S. Touretzky y T.K. Leen (eds.), MIT Press (1995) 625-632
12. Marsland, S., Shapiro, J.; Nehmzow, U.: A self-organising network that grows when required. Neural Networks, 15 (2002) 1041-1058
13. Kaski, S., Lagus, K.: Comparing Self-Organizing Maps. Lecture Notes in Computer Sciences, 1112. Springer, Berlin (1996) 809-814

14. Flórez, F., García, J.M., García, J., Hernández, A.: Hand Gesture Recognition Following the Dynamics of a Topology-Preserving Network. In Proceedings of the 5th IEEE International Conference on Automatic Face and Gesture Recognition, Washington, D.C. IEEE, Inc. (2001) 318-323
15. Villmann, T., Merényi, E., Hammer, B.: Neural maps in remote sensing image analysis. Neural Networks, 16 (2003) 389-403
16. Flórez, F., García, J.M., García, J., Hernández, A.: Analysis of the topology preservation of accelerated Growing Neural Gas in the representation of bidimensional objects. Lecture Notes in Computer Sciences, 3040. Springer, Berlin (2004) 167-176

A Genetic Algorithm for the Shortest Common Superstring Problem

Luis C. González-Gurrola, Carlos A. Brizuela, and Everardo Gutiérrez

Computer Science Department, CICESE Research Center,
Km 107 Carr. Tijuana-Ensenada, Ensenada, B.C., México
{ }
+52–646–1750500

Abstract. This paper[1] presents a genetic algorithm for the shortest common superstring problem. The problem appears in the computational part of a deoxyribonucleic acid (DNA) sequencing procedure as well as in data compression. The proposed genetic algorithm is based on a recently proposed algorithm for the sequencing by hybridization (SBH) problem. The algorithm exploits the crossover operator and modifies the objective function to make the algorithm both more effective and more efficient. Experimental results on real data show the effectiveness of the proposed algorithm.

1 Introduction

Many real world problems can be modeled as the shortest common superstring problem. The problem has several important applications in the areas such as DNA sequencing and data compression.

The shortest common superstring problem can be formulated as follows. Given a set of strings $\mathbf{S} = \{s_1, s_2, ..., s_n\}$ the goal is to find the shortest string N^* such that each $s_i \in \mathbf{S}$ is a string of N^*. This string is known as the shortest common superstring (SCS) of $\mathbf{S}$. Finding the SCS of a set of strings is known to be NP-hard [9]. Many approximation algorithms for the SCS problem have been proposed. One widely used is GREEDY [4] which merges string pairs with maximum overlaps until a single superstring is obtained. Even though this algorithm is conjectured to be a 2-approximation algorithm, Blum and colleagues [4] who were the first to achieve a constant factor approximation for this problem, were able to prove a factor 4 for this problem. They also proposed other variants known as MGREEDY and TGREEDY with approximation factors of 4 and 3, respectively. After these results the approximation factors were reduced to 2.89 by Teng [18]. Czumaj [8] gave a factor of 2.83, Armen achieved a factor of 2.75 and of 2.6 [1, 2], Breslauer gave factors of 2.67 and 2.596 [5]. Up to date the best guarantee is of 2.5 and was achieved by Sweedyk [17].

The only previous work found in the evolutionary computation literature is the one by Zaritsky [19]. He applies a coevolutionary algorithm to the shotgun DNA sequencing which is slightly different from the problem we are dealing

[1] This work is an "in extenso" version of our previous work [11]

with. He also mentions that *their bibliography research revealed no application of evolutionary algorithms to the SCS problem.* In spite of our efforts to search for related work, we were not able to find a single one related to the SCS problem itself. Therefore, we share the point of view of Zaritsky about the existence of previous works.

The remainder of the paper is organized as follows. Section 2 briefly describes the problem we are dealing with. Section 3 explains the methodology proposed to approximately solve the problem. Section 4 presents the experimental setups and results. Finally, section 5 states the conclusions and points out some ideas for future research.

2 Problem Statement

An instance of the SCS problem is represented by a set of strings $\mathbf{S} = \{s_1, s_2, ..., s_i, ..., s_n\} \subseteq \Sigma^*$ over a finite alphabet Σ (e.g. $\Sigma = \{$A, C, G, T$\}$), where Σ^* is the set of all strings constructed from Σ. Without loss of generality let us assume $\mathbf{S}$ to be a free subset, i.e. no string s_i is a substring of s_j for all $i \neq j$. A feasible solution for this problem, a superstring of $\mathbf{S}$, is a string *N,* which contains each string $s_i \in \mathbf{S}$ as a substring. The objective is to find a superstring N^* with the shortest length $|N^*|$ over all superstrings that can be generated.

The SCS problem can be modeled as a maximization problem with respect to the sum of overlaps. For two strings s_i and s_j, the $overlap(s_i, s_j)$ is the longest suffix of s_i that is a prefix of s_j. The problem can be also modeled as a minimization problem with respect to the sum of prefix lengths, where $pref(s_i, s_j)$ is the chain remained after removing the $overlap(s_i, s_j)$ from s_i [18]. The input for what is known as the asymmetric traveling salesman problem (ATSP) is given by a complete digraph *G = (V,A)* and a non negative cost function $c(u,v)$ associated to each arc $< u, v > \in A$ (in general $c(u,v) \neq c(v,u)$). The problem is to find a Hamiltonian cycle in *G* (i.e. a permutation of vertex of *G*) of minimum cost. If each string $s_i \in \mathbf{S}$ is associated to a vertex $v_i \in V$ of G and the arcs costs $c(v_i, v_j)$ are defined by $|pref(s_i, s_j)|$ then finding a minimum cost tour in G is the same as finding the shortest superstring of $\mathbf{S}$ [15]. The resulting associated graph is denominated distance graph. This implies that the optimum tour cost $c(ATSP^*(G))$ of G represents a lower bound for the optimum of the SCS N^* of S, i.e.:

$$c(ATSP^*(G)) \leq |N^*(\mathbf{S})|.$$

Unfortunately the ATSP is also an NP-hard problem. Fortunately, a lower bound for this problem is the well known Held-Karp [13] bound which is a linear programming relaxation of the integer programming formulation for the symmetric TSP. Following this, we can write the following expression:

$$HK \leq c(ATSP^*(G)) \leq |N^*(\mathbf{S})|.$$

We will use this bound to study the solution quality produced by our algorithm. Johnson et al. [14] argue that this bound is a good surrogate for the optimal solution value.

3 The Genetic Algorithm

In this section the proposed algorithm is described in detail. This algorithm is based on a recently proposed algorithm [6] for the sequencing by hybridization problem. The main difference with [6] lies in the objective function which is explained in the following subsection. In order to understand the algorithm and its components we need to see first how the encoding method works.

3.1 Encoding

Each individual is represented by a permutation of indices of substrings in **S**. Specifically, the adjacency-based coding [12] is used: value i at locus j in the chromosome means that substring i follows substring j. Therefore, feasible solutions are represented by subcycle free permutations, except for a single cycle of length |**S**| which takes into account all the substrings in **S**. Figure 1 shows a feasible individual $\boldsymbol{i1}$=[**4 2 3 0 1**], for a given set **S** = {CTG,ACT,GGA,GAC,TGA}. In this chromosome, the number 4 at locus 0 indicates that the substring at position 0 in **S** (CTG), is followed by the substring at position 4 in **S** (TGA). In this way, following the indices in **S**, the sequence of substrings that this individual represents is: CTG,TGA,ACT,GGA,GAC (0, 4, 1, 2, 3, 0). Notice that this solution defines a set of candidate sequences not a single sequence, as it will become clear when we explain the objective function computation. Once we have understood how to encode the solutions, we can explain the general algorithm.

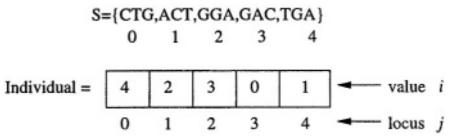


Fig. 1. Adjacency Based Representation for the SCS Problem, |**S**| = 5

Algorithm 1. SCS GA
Input: A set of strings **S**
Output: A candidate superstring N

```
1 for $i = 1$ to $Pop\_Size$
2         do Generate_Individual($i$)
3             Evaluate Objective_Function($i$)
4 do Select individuals to update the population
5     Keep the best individual found
6 for $i = 1$ to $Num\_Of\_Iter$
7         do Crossover with probability $p_c$
8             Apply Generational Replacement
9         for $j = 1$ to $Pop\_Size$
10                do Evaluate Objective_Function($j$) of individual $j$
11        do Select individuals to update the population
12            Keep the best individual found
```

The algorithm input is given by a set of substrings $\mathbf{S}$ with cardinality $|\mathbf{S}|$. *Pop_size* is the GA population size. The function $\mathsf{Generate\_Individual}(i)$ randomly and uniformly generates individual i. Subcycles must be avoided in the individuals to ensure the use of all substrings in $\mathbf{S}$. Evaluate $\mathsf{Objective\_Function}(i)$ computes the fitness for each individual i, we propose to compute the fitness function as follows. The fitness of each individual is the maximum number k of overlapping characters over all the starting positions in the chromosome, and at each position all substrings must be considered. Figure 2 shows the way the objective function is computed. First, we start at position zero and the resulting cycle is (0,4,1,2,3,0). Substring 0 (CTG) overlaps with substring 4 (TGA) in two characters: TG, at this point $k = 2$. Then, we look at substring 1 (ACT) which overlaps in one character with substring 4: A, then $k = 3$. We continue with substring 2 (GGA), this one does not overlap with substring 1, then $k = 3$ remains unchanged. Finally, substring 3 (GAC) overlaps in two characters with substring 2, therefore, the final value for k is five $(k = 5)$. We repeat this process starting at each position in the chromosome, this can be done given that the permutation of indices in the individual generates a cycle of length $|\mathbf{S}|$. After having counted the overlapping characters starting at each position in the chromosome, we choose the starting position with the maximum number of overlapping characters the individual represents. For this example, this number is generated starting at position 2 (2,3,0,4,1,2), $k = 6$. The corresponding string is GGACTGACT with $n = 9$, the minimum length superstring of all the superstrings that this individual can represent. For each individual, its fitness value is normalized with respect to the maximum number of overlapping characters obtained when all substrings overlap in $l-1$ characters, i.e. when $K = ((|\mathbf{S}|-1)(l-1))$, where l is the length of the substrings. In [6], the fitness of each individual is the maximum number of substrings, that, maximally overlapped, generates a sequence whose length must not be longer than the original sequence length.

The individuals are selected (steps 4 and 11) according to the stochastic remainder method without replacement [10]. The population for the next generation is constructed based on the already selected individuals, which are randomly paired to undergo crossover. By applying generational replacement with elitism, the new population will replace the old one, always keeping the best individual (steps 5 and 12). The stopping criterion is given by a maximum number of iterations, *Num_Of_Iter* (step 6).

3.2 Crossover Operator

The crossover operator was proposed in the authors recent work [6] as an improvement over the one proposed by Blazewicz et al. [3], originally for the DNA sequencing problem. In [3] the selection of substrings for constructing new individuals is based on a probabilistic choice. In [6], the best successor in the parents is always selected as long as a subcycle is not generated. Otherwise, the best successor not generating a subcycle, is selected among the remaining substrings in $\mathbf{S}$. Details of the operator are supplied below.

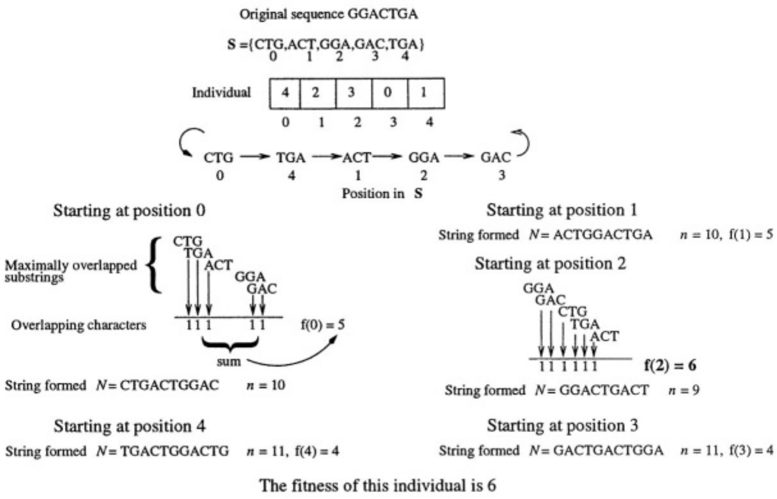

Original sequence GGACTGA
S ={CTG,ACT,GGA,GAC,TGA}
0 1 2 3 4
Individual
4 2 3 0 1
0 1 2 3 4
CTG → TGA → ACT → GGA → GAC
0 4 1 2 3
Position in S
Starting at position 0
Maximally overlapped substrings
Overlapping characters 111 11 f(0) = 5
sum
String formed N= CTGACTGGAC n = 10
Starting at position 1
String formed N= ACTGGACTGA n = 10, f(1) = 5
Starting at position 2
111111 f(2) = 6
String formed N= GGACTGACT n = 9
Starting at position 4
String formed N= TGACTGGACTG n = 11, f(4) = 4
Starting at position 3
String formed N= GACTGACTGGA n = 11, f(3) = 4
The fitness of this individual is 6

Fig. 2. Fitness Function Computation for Individual $i1$=[4 2 3 0 1] and $\mathbf{S}$ = {CTG,ACT,GGA,GAC,TGA}, $f(i1) = 6$

```
Algorithm 2. Crossover
Input: Two subcycle free permutations
Output: A subcycle free permutation
1 set the first substring randomly (for child 1)
2 for $i = 1$ to $|\mathbf{S}| - 2$
3         if it does not produce subcycle
4             do pick up the best successor in the parents of the chromosome
5         otherwise
6                 do pick up the best overlapping substring from S such that
                     a subcycle is not introduced
7 set the last substring
```

Figure 3 shows how this operation is performed. Suppose again that the set of substrings is given by $\mathbf{S}$={CTG,ACT,GGA,GAC,TGA}. The first substring randomly selected is GAC, substring 3, and its randomly selected locus is 2 (Fig. 3A). Notice that for completing the cycle, substring 2 will be the last one to be selected in the chromosome. We look at parent 1 and parent 2 for the successors of substring 3, which are 4 and 0, respectively (Fig. 3B). The best successor (best overlapping substring) is substring 0 (Fig. 3C). Next, we look at the parents for the best successor of substring 0, which is substring 2, but this substring generates a subcyle (Fig. 3D), so we look at $\mathbf{S}$ for the best successor which is substring 4 (Fig. 3E). For substring 4, both successors in the parents generate a subcycle, so we look at $\mathbf{S}$, given that substring 2 will be the last one, we have only one option, substring 1 (Fig. 3F). Finally, for completing the cycle, we set substring 2 (Fig. 3G).

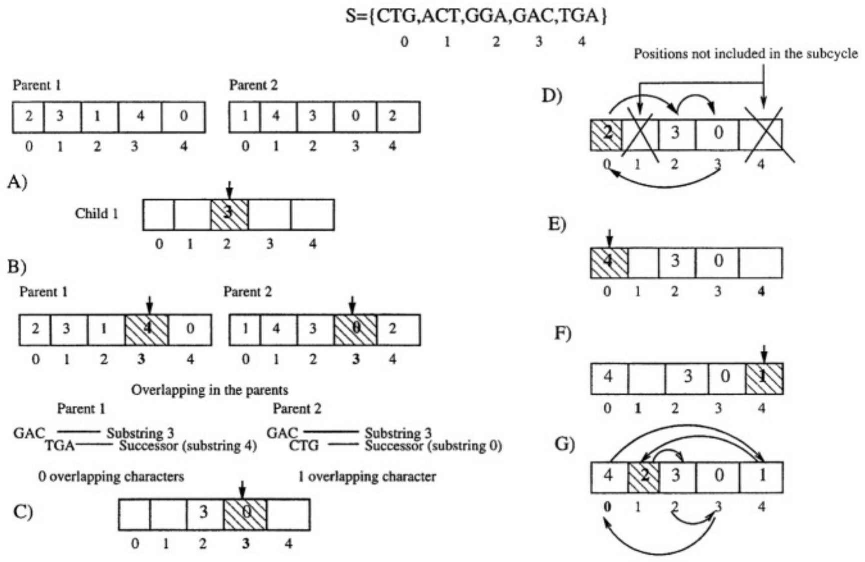


Fig. 3. Almost Deterministic Greedy Crossover for **S** = {CTG,ACT,GGA,GAC,TGA}

4 Experimental Setup and Results

As we mention before, the SCS approximately models the DNA sequencing problem as well as the data compression problem. We have decided to deal with a sequencing problem because of the many available benchmarks coming from the real world. All sets **S** used in the experiment have been derived from DNA sequences coding human proteins (taken from GenBank, National Institute of Health, USA). Their accession numbers are given by D90402, X02160, X07994, X51841, Y00062, Y00093, Y00649, X14034, D13720 and X01060, respectively. Each derived instance was modified with the elimination (randomly erased) of 20% of the substrings. Given that $100 \leq |\mathbf{S}| \leq 3000$, from 25 (for $|\mathbf{S}| = 100$) to 750 (for $|\mathbf{S}| = 3000$) substrings are missing in the original set of substrings (e.g. for generating instances of $|\mathbf{S}| = 3000$, 750 randomly selected substrings were erased from the original set of size 3750). The length of substrings in all cases is l=10. For each case, 10 different instances were generated, the parameters for the instances are shown in Table 1. Parameters for the algorithm have been set to Pop_Size = 50, Num_Of_Iter = 20, and p_c = 1.0. Since this algorithm has a strong greedy component, an increment in the number of iterations will not improve significantly the results. All the experiments have been performed on a PC with Athlon XP 2.0 GHz processor, 512 MB of RAM and Linux Mandrake 9.1 operating system.

The criterion for measuring the solution quality is, of course, the distance from the optimal solution, typically measured as the percentage by which the algorithm solution's length exceeds the optimum solution length. In order to use this standard, one must know the optimal solution value. In order to provide a

Table 1. The instances generated for the experiments and the corresponding HK lower bound

Sequence's original length	|S|	Held-Karp bound
134	100	133.2
259	200	258.4
384	300	383
509	400	508.2
634	500	633.1
759	600	757.5
884	700	882.3
1009	800	1007.1
1134	900	1131.6
1259	1000	1255.4
2559	2000	2506.1
3759	3000	3758

point of reference for the optimal length of each instance, we compute its Held-Karp (HK) lower bound on the optimal solution [13]. The SCS GA was applied to each generated instance. For each instance the SCS GA was executed 30 times. Table 2 presents the approximation ratio between the obtained solution and its HK bound. Column 2 presents average values for 10 instances over 30 runs each. Column 3 presents the standard deviation, which is shown as percentage over the results in column 2. As it is argued by Sipper [16] what ultimately counts in problem solving by an evolutionary algorithm is the best result. Column 4 presents average values of the best results obtained for each instance, the last column shows its standard deviation.

As we can see, the results of the SCS GA are close to the Held-Karp lower bound. The best solutions lengths (except for the case of $|\mathbf{S}| = 3000$) exceeds the

Table 2. Approximation rate for the SCS GA with respect to the Held-Karp lower bound

|S|	SCS GA	S.D. (%)	Best	S.D. (%)
100	1.054	2.7	1.020	1.97
200	1.038	1.34	1.008	1.08
300	1.033	1.07	1.012	1.28
400	1.025	0.84	1.008	0.59
500	1.026	0.86	1.012	0.58
600	1.023	0.86	1.010	0.63
700	1.024	0.99	1.015	0.8
800	1.023	0.81	1.014	0.64
900	1.020	0.78	1.011	0.49
1000	1.020	0.67	1.012	0.5
2000	1.021	0.73	1.012	0.5
3000	1.032	1.15	1.021	0.9

Table 3. Similarity percentage of the solutions and the original sequences

Instance Size |S|	Similarity percentage score
100	98.95%
200	96.48%
300	97.16%
400	96.46%
500	87.85%
600	83.39%
700	70.58%
800	70.72%
900	69.50%
1000	57.18%
2000	31.22%
3000	29.20%

HK bound in no more than 1.5%. Even when $2000 \leq |\mathbf{S}| \leq 3000$, the length of solutions are not far from the HK bound. The standard deviation of the results shows the uniformity in the solutions the GA achieves.

It would also be interesting to see how similar the solutions are to the original DNA sequences, from where we derived the sets of substrings. For this purpose, we compared the solutions generated by the GA with the original sequence using a pairwise alignment algorithm. Table 3 presents how similar the solutions generated by the GA and the original sequences are, if both sequences are identical, then the similarity score is 100%. From Table 3 we can observe that the similarity score for the first five sets of instances, where $100 \leq |\mathbf{S}| \leq 500$, shows high similarity with respect to the original sequence, and this point is of special interest given that for real hybridization experiments $100 \leq |\mathbf{S}| \leq 500$ [7]. On the other hand, as the instance size grows beyond $|\mathbf{S}| = 500$, the similarity score decreases below an acceptable level. A reason for this behavior could be explained with a conjecture saying that the number of superstrings with minimal length increases with $|\mathbf{S}|$. However, the optimal sequence is 100% similar to only one of those superstrings.

Figure 4 shows the average computation times needed by the GA for obtaining the results shown before. The abrupt increment shown in the computing time is due to the increment in the size of the instances (from 100 increments to 1000 increments) we are dealing with.

5 Conclusions

An efficient and effective genetic algorithm for the SCS problem has been proposed. The procedure is based on a recently proposed algorithm for the sequencing by hybridization problem. We propose a new fitness function which exploits the problem structure. Experimental results carried out on real DNA data show that the proposed algorithm can work effectively on a set of specific instances.

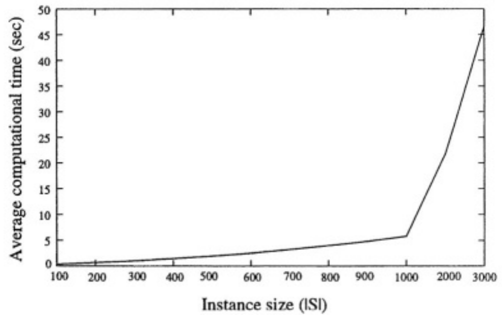


Fig. 4. Average computational time for all instances

Relative errors less than 6% from the optimum tell us about the suitability of this algorithm for real world problems. As a future research we will further explore the rationale for success of the algorithm as well as its suitability for other SCS application problems. Also, we are very interested in the effects of the implicit mutation which occurs into the crossover operator when none of the parents genes are selected. Finally, we would like to analyze how to improve this algorithm regarding the similarity score for large instances.

Acknowledgments. This research has been partially supported by Laboratorio Franco-Mexicano de Informática 2003. The authors would like to thank Jazmín Romero for her valuable help in computing the HK bounds.

References

1. C. Armen and C. Stein. Improved Length Bounds for the Shortest Superstring problem. In *Proceedings of the 4th Workshop on Algorithms and Data Structures,* volume 955 of *LNCS,* pages 494–505. Springer-Verlag, 1995.
2. C. Armen and C. Stein. A $2\frac{2}{3}$ approximation algorithm for the shortest superstring problem. In *Proceedings of Combinatorial Pattern Matching,* volume 1075 of *LNCS,* pages 87–101. Springer-Verlag, 1996.
3. J. Blazewicz, M. Kasprzak, and W. Kuroczycki. Hybrid Genetic Algorithm for DNA Sequencing with Errors. *Journal of Heuristics,* 8:495–502, 2002.
4. A. Blum, T. Jiang, M. Li, J. Tromp, and M. Yannakakis. Linear approximation of shortest superstring. In *Proceedings of the 23rd ACM Symp. on theory of computing,* 1991.
5. D. Bresaluer, T. Jiang, and Z. Jiang. Rotations of Periodic String and Short Superstring. *Journal of Algorithms,* 24:340–353, 1996.
6. Carlos A. Brizuela, Luis C. González, and Heidi J. Romero. An Improved Genetic Algorithm for the Sequencing by Hybridization Problem. In *Proceedings of Applications of Evolutionary Computing, LNCS,* volume 3005, pages 11–20, 2004.
7. P.A. Caviani, D. Solas, E.J. Sullivan, M.T. Cronin, C.P. Holmes, and S.P.A. Fodor. Light-Generated Oligonucleotide Arrays for Rapid DNA Sequence Analysis. *Proceedings of the National Academy of Sciences of the USA 91,* 91:5022–5026, 1994.

8. A. Czumaj, L. Gasieniec, M. Piotrow, and W. Rytter. Sequential and Parallel Approximation of Shortest Superstrings. *Journal of Algorithms,* 23:74–100, 1995.
9. M. Garey and D. Johnson. *Computers and Intractability: A guide to the theory of NP-Completeness.* Freeman, 1978.
10. D.E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, 1989.
11. Luis C. González, Heidi J. Romero, and Carlos A. Brizuela. A Genetic Algorithm for the Shortest Common Superstring Problem. In *Proceedings of the Genetic and Evolutionary Computation Conference, LNCS,* volume 3103, pages 1305–1306. LNCS, 2004.
12. J. Grefenstette, R. Gopal, B. Rosmaita, and D. Van Gucht. Genetic algorithms for the travelling salesman problem. In *Proceedings of the first International Conference on Genetic Algorithms and Application,* volume 1, pages 160–168, 1985.
13. M. Held and R. Karp. The traveling salesman problem and minimum spanning trees. *Operational Research,* 18:1138–1182, 1970.
14. D.S. Johnson, L.A. McGeoch, and E.E. Rothberg. Asymptotic experimental analysis for the Held-Karp traveling salesman bound. *In Proceedings of the 7th Annual ACM-SIAM Symposium in Discrete Algorithms,* pages 341–350, 1996.
15. P.A. Pevzner. *Computational Molecular Biology: An Algorithmic Approach.* MIT Press, 2000.
16. M. Sipper. A succes story or an old wives' tale? On judging experiments in evolutionary computation. *Complexity,* 4(5):31–33, 2000.
17. E. Sweedyk. A $2\frac{1}{2}$ approximation algorithm for the shortest common superstring,University of California, Ph.D. Dissertation, 1995.
18. S.H. Teng and F. Yao. Approximating shortest superstrings. In *Proceedings of the 34th IEEE Symp. Found. Comp. Science,* pages 158–165, 1993.
19. A. Zaritsky. Coevolving Solutions to the Shortest Common Superstring Problem. Master's thesis, Departament of Computer Science, Ben-Gurion University, Beer-Sheva 84105, Israel, 2003.

Improving the Efficiency of a Clustering Genetic Algorithm

Eduardo R. Hruschka, Ricardo J. G. B. Campello, and Leandro N. de Castro

Catholic University of Santos (UniSantos),
R. Carvalho de Mendonça, 144,
CEP 11070-906, Santos/SP – Brazil
{erh, campello, lnunes}@unisantos.br

Abstract. Finding optimal clusterings is a difficult task. Most clustering methods require the number of clusters to be specified in advance, and hierarchical methods typically produce a set of clusterings. In both cases, the user has to select the number of clusters. This paper proposes improvements for a clustering genetic algorithm that is capable of finding an optimal number of clusters and their partitions automatically, based upon numeric criteria. The proposed improvements were designed to enhance the efficiency of a clustering genetic algorithm. The modified algorithms are evaluated in several simulations.

1 Introduction

Clustering is a task in which one seeks to identify a finite set of categories (clusters) to describe a given data set, both maximizing homogeneity within each cluster and heterogeneity among different clusters [1]. In other words, objects that belong to the same cluster should be more similar to each other than objects that belong to different clusters. Thus, it is necessary to devise means of evaluating the similarities among objects. This problem is usually tackled indirectly, i.e. distance measures are used to quantify the dissimilarity between two objects, in such a way that more similar objects have smaller dissimilarity measures. Several dissimilarity measures can be employed for clustering tasks, such as the commonly used Euclidean distance.

There are three main types of clustering techniques: overlapping, hierarchical and partitioning. This work considers that clustering involves the partitioning of a set $\mathbf{X}$ of objects into a collection of mutually disjoint subsets $\mathbf{C}_i$ of $\mathbf{X}$. Formally, let us consider a set of N objects $\mathbf{X}=\{\mathbf{x}_1,\mathbf{x}_2,...,\mathbf{x}_N\}$ to be clustered, where each $\mathbf{x}_i \in \Re^\rho$ is an attribute vector consisting of ρ real-valued measurements. The objects must be clustered into non-overlapping groups $\mathbf{C}=\{\mathbf{C}_1,\mathbf{C}_2,...,\mathbf{C}_k\}$ where k is the number of clusters, such that:

$$\mathbf{C}_1 \cup \mathbf{C}_2 \cup ... \cup \mathbf{C}_k = \mathbf{X}, \quad \mathbf{C}_i \neq \varnothing, \quad \text{and} \quad \mathbf{C}_i \cap \mathbf{C}_j = \varnothing \text{ for } i \neq j. \tag{1}$$

Many methods described in the literature assume that the value k is given by the user [2]. These methods search for k clusters according to a predefined criterion. By doing so, the number of ways of classifying N objects into k clusters is [3]:

$$\mathrm{NW}(N,k) = \frac{1}{k!}\sum_{i=0}^{k}(-1)^i\binom{k}{i}(k-i)^N \tag{2}$$

Thus, it is not easy to find the best clustering even assuming that k is known. Indeed, this is rarely the case in practice. An usual approach is to run a clustering algorithm several times and, based on these results, choose the value for k that provides the most *natural* clustering. This strategy assumes domain knowledge and usually has the disadvantage of searching for the best solution in a small subset of the search space. Consequently, these methods have low probabilities of success. Another alternative involves optimizing k according to numeric criteria. In this case, k is unknown and the number of ways of grouping N objects into k clusters, considering n different scenarios (each one resulting from a different value of k), is [3]:

$$\sum_{i=1}^{n} NW(N,k) \tag{3}$$

In other words, the problem of finding an optimal solution to the partition of N data into k clusters is NP-complete [4] and, provided that the number of distinct partitions of N objects into k clusters increases approximately as $k^N/k!$, attempting to find a globally optimum solution is usually not computationally feasible [1]. This difficulty has stimulated the search for efficient approximation algorithms.

Genetic algorithms are widely believed to be effective on NP-complete global optimization problems and they can provide good sub-optimal solutions in reasonable time [4]. Under this perspective, a genetic algorithm specially designed for clustering problems was introduced in [5]. Although this algorithm, called Clustering Genetic Algorithm (CGA), has provided encouraging results on a representative number of datasets [5], its efficiency can be further improved, as shown in the present paper.

The paper is organized as follows. Section 2 reviews the CGA [5], whereas Section 3 describes the proposed improvements. Section 4 reports the results of simulations performed in a dataset usually employed as a benchmark for clustering problems. In the fifth section we present our conclusions, based on statistical analyses of the simulation results, and point out some future work.

2 Review of the Clustering Genetic Algorithm (CGA)

2.1 Encoding Scheme

The CGA [5] is based on a simple encoding scheme. Let us consider a data set formed by N objects. Then, a genotype is an integer vector of $(N+1)$ positions. Each position corresponds to an object, i.e., the i-th position (gene) represents the i-th object, whereas the last gene represents the number of clusters (k). Thus, each gene has a value over the alphabet $\{1,2,3,...,k\}$. For instance, in a dataset composed of 20 objects, a possible genotype is:

Genotype 1: 22345123453321454552 5.

In this case, five objects $\{1,2,7,13,20\}$ form the cluster whose label is 2. The cluster whose label is 1 has 2 objects $\{6,14\}$, and so on. Finally, the last gene corresponds to the number of clusters.

Standard genetic operators may not be suitable for clustering problems for several reasons [6,5]. First, the encoding scheme presented above is naturally redundant, i.e., the encoding is *one-to-many*. In fact, there are $k!$ different genotypes that represent the same solution. For example, there are 5! different genotypes that correspond to the same clustering solution represented by *Genotype* 1. Some of them are:

Genotype 2: 33451234514432515113 5
Genotype 3: 44512345125543121224 5
Genotype 4: 55123451231154232335 5
Genotype 5: 11234512342215343441 5

Thus, the size of the search space the genetic algorithm has to search is much larger than the original space of solutions. This augmented space may reduce the efficiency of the genetic algorithm. In addition, the redundant encoding also causes the undesirable effect of casting context-dependent information out of context under the standard crossover, i.e., equal parents can originate different offspring. For example, if *Genotype* 2 was crossed-over with *Genotype* 3, the resulting genotypes should be equal to their parents, since they represent the same solution to the clustering problem. Unfortunately, however, that rarely happens. In order to make this situation more easily observed, let us apply a two-point crossover, indicated by two vertical lines, to the following genotypes, which represent the same clustering solution:

Genotype 6: 2 | **2222** | 111113333344444 4
Genotype 7: 4 | **4444** | 333335555511111 4

The genotypes of the resulting offspring are given by:

Offspring 1: 2**4444**111113333344444 4
Offspring 2: **42222**333335555511111 5

These offspring are different from their parents; that is, they represent different clustering solutions. In addition, *offspring 2* represents five clusters, whereas their parents represent just four.

An alternative to solve these problems is to employ a renumbering algorithm. This is not enough, however, because traditional genetic algorithms usually do not perform well when applied to clustering tasks, since they manipulate gene values without taking into account their connections with other genes. Indeed, the interconnections among gene values (represented by cluster labels) constitute the genuine optimization goal in clustering problems. For this reason, the development of genetic operators specially designed to clustering problems has been investigated [6,5]. In this context, the CGA operators proposed in [5] are of particular interest since they operate on constant length chromosomes. These cluster-oriented (i.e. context-sensitive) operators are described in Sections 2.2 and 2.3.

2.2 Crossover

The CGA crossover operator combines clustering solutions coming from different genotypes. It works in the following way. First, two genotypes (**A** and **B**) are selected.

Then, assuming that **A** represents k_1 clusters, the CGA randomly chooses $c \in \{1,2,\ldots,k_1\}$ clusters to copy into **B**. The unchanged clusters of **B** are maintained and the changed ones have their objects allocated to the corresponding nearest clusters (according to their centroids, i.e. mean vectors). This way, an offspring **C** is obtained. The same procedure is used to get an offspring **D**, but now considering that the changed clusters of **B** are copied into **A**. To illustrate it, let us consider the following genotypes, in which the last gene (number of clusters) is not represented:

A – 1123245125432533424
B – 1212332124423221321

For example, let us assume that clusters 2 and 3 of **A** (below in boldface) were randomly selected:

A – 11**232**451**2**54**32**5**33**4**2**4

When these clusters are copied into **B** they modify clusters {1,2,3} of **B**, whereas cluster 4 is not modified:

$\mathbf{B} - \underline{12}\mathbf{232}\underline{321}\mathbf{2}44\mathbf{32}\underline{2}\mathbf{33}\underline{3}\mathbf{2}\underline{1}$

The underlined positions, which correspond to those genes indirectly affected by clusters 2 and 3 of **A**, are now changed to 0 (zero):

C – 0023200024432033020

The genes set equal to zero will then be placed into the nearest clusters (according to their centroids) already encoded by **C**. The same procedure is employed to get an offspring **D**, except for the selected clusters to be copied into **A**. These clusters are now the clusters of **B** initially changed for deriving offspring **C**, namely {1,2,3}. Thus, the crossover operator produces offspring usually formed by a number of clusters that are neither smaller nor larger than the number of clusters of their parents.

2.3 Mutation

Two operators for mutation are used in the CGA. The first operator works only on genotypes that encode more than two clusters. It eliminates a randomly chosen cluster, placing its objects to the nearest remaining clusters (according to their centroids). The second operator divides a randomly selected cluster into two new ones. The first cluster is formed by the objects closer to the original centroid, whereas the other cluster is formed by those objects closer to the farthest object from the centroid.

2.4 Objective Function

The objective function is based on the silhouette [2]. To explain it, let us consider an object i belonging to cluster **A**. So, the average dissimilarity of i to all other objects of **A** is denoted by $a(i)$. Now let us take into account cluster **C**. The average dissimilarity of i to all objects of **C** will be called $d(i,\mathbf{C})$. After computing $d(i,\mathbf{C})$ for all clusters $\mathbf{C} \neq \mathbf{A}$, the smallest one is selected, i.e. $b(i) = \min d(i,\mathbf{C})$, $\mathbf{C} \neq \mathbf{A}$. This value represents the dissimilarity of i to its neighbor cluster, and the silhouette $s(i)$ is given by:

$$s(i) = \frac{b(i) - a(i)}{\max\{\, a(i), b(i)\}} \qquad (4)$$

It is easy to verify that $-1 \leq s(i) \leq 1$. Thus, the higher $s(i)$ the better the assignment of object i to a given cluster. In addition, if $s(i)$ is equal to zero, then it is not clear whether the object should have been assigned to its current cluster or to a neighboring one [7]. Finally, if cluster **A** is a singleton, then $s(i)$ is not defined and the most neutral choice is to set $s(i) = 0$ [2]. The objective function is the average of $s(i)$ over $i = 1,2,...,N$ and the best clustering is achieved when its value is as high as possible.

2.5 Selection, Settings and Initial Population

The genotypes corresponding to each generation are selected according to the roulette wheel strategy [8], which does not admit negative objective function values. For this reason, a constant equal to one is summed up to the objective function before the selection procedure takes place. In addition, the best (highest fitness) genotype is always copied into the succeeding generation, i.e., an elitist strategy is adopted [8].

The CGA does not employ crossover and mutation probabilities; that is, the designed operators are necessarily applied to some selected genotypes after the roulette wheel selection procedure is performed. Particularly, it is assumed that 50% of the selected genotypes are crossed-over, 25% are mutated by Operator 1 and 25% are mutated by Operator 2.

The initial population for the CGA is randomly generated. Specifically, for each genotype the number of clusters k is drawn from the set {2,3,...,NC}, where NC is the (previously defined) maximum possible number of clusters. Then, each gene takes a random value from the set $\{1,2,..., k\}$.

3 Enhancing the CGA

The main steps of the CGA are summarized in Fig. 1, which will be used as a basis to explain the modifications proposed to improve the CGA efficiency. Basically, three incremental modifications are introduced: (i) include the k-means algorithm [7,9] as a local search engine; (ii) apply a dissimilarity measure based on centroids; and (iii) use a more sensitive objective function. The *original* CGA [5] will be termed CGA-I and the modified algorithms, to be detailed in the next subsections, will be called CGA-II, CGA-III and CGA-IV, respectively.

```
1. Initialize a population of random genotypes;
2. Evaluate each genotype according to its silhouette;
3. Apply a linear normalization;
4. Select genotypes by proportional selection;
5. Apply crossover and mutation;
6. Replace the old genotypes by the ones formed in step 5;
7. If convergence is attained, stop; else, go to step 2.
```

Fig. 1. Main steps of the Clustering Genetic Algorithm (CGA-I)

3.2 CGA-II

Clustering algorithms involving the calculation of the mean (centroid) of each cluster are often referred to as *k*-means algorithms [2,9,7]. Basically, these algorithms partition a dataset of *N* objects into *k* clusters. The *k* value, which represents the number of clusters, is specified by the user. These algorithms try to minimize the sum of distances (e.g. Euclidean) between objects of a cluster and its centroid. In this work, we consider the *k*-means algorithm depicted in Fig. 2:

```
1. Generate a random initial partition of objects into $k$ nonempty clusters;
2. Compute the cluster centroids (mean vectors) of the current partition;
3. Assign each object to the cluster with the nearest centroid;
4. If the convergence criterion has been satisfied, then stop; else, go to step 2.
```

Fig. 2. Main steps of the *k*-means algorithm

The convergence criterion can be defined either as the maximum number of iterations *(t)* of steps 2 and 3 or as the maximum absolute difference between centroids in two consecutive iterations.

The *k*-means algorithm has two main drawbacks: (i) it may get stuck at suboptimal centroids; and (ii) the user has to provide the number of clusters *(k)*. On the one hand, these problems are lessened by CGA-I, which can find the best clustering in a dataset in terms of both the number of clusters and centroids. As described in Section 2, the crossover and mutation operators of CGA-I are suitable for clustering problems because they merge, split, and eliminate clusters, thus trying to find better solutions, which, in turn, can represent better *initial* centroids for the *k*-means algorithm. On the other hand, the *k*-means also improves CGA-I since it helps the genetic algorithm to work with solutions that are at least locally optima. Besides, the *k*-means algorithm can find less than *k* clusters when, in a set of *k* centroids, at least one is farther from all the objects than all the others. This feature also helps CGA-II to focus on the most promising solutions, decreasing *k* when necessary.

```
1. Initialize a population of random genotypes;
2. Apply the $k$-means algorithm to each genotype;
3. Evaluate each genotype according to its silhouette;
4. Apply a linear normalization;
5. Select genotypes by proportional selection;
6. Apply crossover and mutation;
7. Replace the old genotypes by the ones formed in step 6;
8. If the convergence is attained, stop; else, go to step 2.
```

Fig. 3. Main steps of CGA-II (CGA-I + *k*-means)

In summary, we propose a hybrid method that makes use of the advantages of both techniques. As can be seen in Fig. 3, a new step, which refers to the application of *k*-means (Fig. 2), was inserted into the original CGA-I (Fig. 1). From now on, we will refer to the algorithm described in Fig. 3 as CGA-II. The computational cost of CGA-I is estimated as $O(N^2)$, whereas the cost of the *k*-means algorithm is $O(tkN)$; usually $\{t,k\} << N$. Thus, in principle, CGA-II is as costly as CGA-I. However, the use of *k*-means together with the standard CGA can speed up the CGA-I convergence.

3.2 CGA-III

The second modification proposed concerns improving the efficiency of the CGA-II objective function. In this sense, the traditional silhouette function [2], which depends on the computation of all distances among all objects, is replaced with one based on the distances among the objects and the centroids of the clusters currently available. The parameter $a(i)$ of Eqn. (4) now corresponds to the dissimilarity of object i to its corresponding cluster centroid (**A**). Thus, it is necessary to compute only one distance to have the $a(i)$ value, instead of calculating all the distances between i and the other objects of **A**. Similarly, instead of computing $d(i,\mathbf{C})$ as the average dissimilarity of i to all objects of **C**, $\mathbf{C} \neq \mathbf{A}$, we propose to compute the distances between i and the **C** centroid. These modifications significantly reduce the computational cost of the algorithm, which depends strongly on the calculus of the objective function. Furthermore, these modifications cause a synergy between the objective function and the *k*-means algorithm to take place, for they are now both based on centroids. Finally, an additional advantage obtained with this modification comes from the fact that it is no longer necessary to store any dissimilarity matrix. In summary, CGA-III is different from CGA-II in step 3, which makes the CGA-III complexity to be $O(N+tkN)$. As previously observed, usually $\{t,k\} << N$. In these cases, CGA-III is $O(N)$.

3.3 CGA-IV

In order to evaluate the robustness of CGA-III, we also propose an alternative objective function inspired by ideas from the original silhouette concept. More specifically, we now employ Eqn. (5) (instead of Eqn. (4)) to calculate $s(i)$:

$$s(i) = \frac{b(i)}{a(i) + \varepsilon}, \tag{5}$$

where $b(i)$ and $a(i)$ are the same as in CGA-III. As before, this equation favors compact and well-separated clusters, i.e. $a(i)$ near zero and $b(i)$ as high as possible. The term ε is necessary to compute $s(i)$ when $a(i)$ is zero, i.e. when all objects of cluster **A** are equal to one another. This modified objective function seems to be more sensitive to slight changes in $a(i)$ and $b(i)$, which in turn may correspond to significant changes in the clustering solution.

4 Performance Evaluation

The proposed modifications to the original CGA were evaluated by means of several simulations performed using the Ruspini Dataset [2], which is a benchmark to evaluate clustering methods. In this dataset, there are 75 objects described by means of two features {x, y}. Four groups, formed by 20, 23, 17 and 15 objects each, represent the *right* clustering, as shown in Fig. 4.

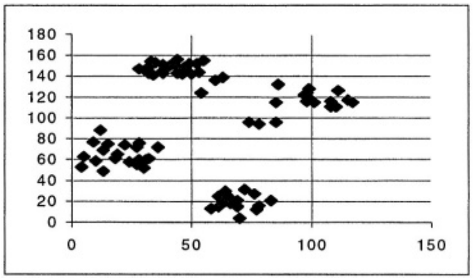


Fig. 4. Ruspini dataset

The algorithms CGA-{I, II, III and IV} were implemented in *C Language, incrementally* from CGA-I to CGA-IV and trying to use just the strictly necessary additional commands. This way, reasonably fair comparisons, in terms of efficiency, can be performed. For each of these algorithms, simulations were run in a Pentium IV, 1.8 GHz CPU, 512 MB RAM, until the objective function value corresponding to the right solution is first met. The computing time necessary to perform such tasks was stored.

In each simulation, a population formed by four genotypes was used. Thus, the crossover operator was employed in two of them, and the other ones were mutated (either by operator 1 or 2). This is the smallest possible population in the context of our algorithms (see Section 2.5 for further details). By doing this, we can show how effective the proposed methods are in the most adverse situation.

The Euclidean norm was used both in the CGAs and in the *k*-means. In addition, the data were normalized within the interval [0,1]. Finally, the *k*-means algorithm was programmed to stop when one of the following criteria is satisfied: (i) five iterations have been completed; or (ii) the maximum absolute difference between centroids in two consecutive iterations is less than or equal to 0.001.

Initially, we performed simulations considering initial populations randomly generated, in which each genotype has a number of clusters taken from the set {2,…,37}, which represents the possible number of clusters with at least 2 objects in the Ruspini dataset. Although one could argue that 75 is the maximum number of clusters, this is the trivial solution for a silhouette-based clustering problem, and it does not have a practical meaning. In addition, assuming that the initial number of clusters belongs to the set {2,…,N/2} is a reasonable approach when one has no idea about the *natural* number of clusters. In all simulations, the proposed algorithms found the right clusters (Fig. 4). A summary of the simulation results for each algorithm is described in Table 1, where μ and σ respectively represent the average and the standard deviation over 35 simulations and NG stands for the number of generations for convergence.

Table 1. Results obtained for initial populations randomly generated - $k_{\text{initial}} \in \{2,\ldots,37\}$

Algorithm	μ-CPU Time (s)	σ-CPU Time (s)	μ-NG	σ-NG
CGA-I	4.2243	10.9430	2058.66	6181.83
CGA-II	0.2386	0.3837	26.11	45.44
CGA-III	0.0706	0.0909	8.03	13.72
CGA-IV	0.0691	0.1069	6.66	12.33

Table 1 indicates that increasingly better results in terms of reduction in CPU time were obtained from CGA-I to CGA-IV. In fact, this hypothesis was statistically tested. To do so, we applied the Wilcoxon/Mann-Whitney test [10], which is a non-parametric (distribution-free) test used to compare two independent data samples - the samples are assumed independent because the initial populations, which characterize the starting point to the evolutionary search, were randomly generated. In summary, we concluded that, at the 5% significance level, our simulations support that the average CPU times of CGA-II are less than those of CGA-I, as well as the CPU times of CGA-III are less than those of CGA-II When the CGA-IV performance is compared with that of CGA-III, however, there is no sample evidence to support that the former is better. For illustrative purposes, Table 1 also indicates that the *number of generationsfor convergence* is correlated with the *CPU time.* However, the same number of generations, in different simulations, can lead to different CPU times, because of the probabilistic nature of an evolutionary search.

The simulation results obtained for initial populations randomly generated with $k \in \{2,\ldots,37\}$ are very encouraging, because they have a practical appeal. However, we also decided to evaluate the performance of the CGAs in two *extreme* situations in terms of the number of clusters, namely, choosing initial populations in which all genotypes represent 2 clusters (minimum amount) or 37 clusters (maximum amount). This way, the statistical evaluation becomes more focused on the algorithms, in the sense that the additional variability represented by the initial population is ruled out. Although these scenarios do not have a practical appeal, they can be useful for a better evaluation of the performance of the proposed methods. Table 2 shows the simulation results in terms of CPU time for 35 simulations in each scenario.

Table 2. Simulation results for initial populations randomly generated (2 and 37 clusters)

	Initial number of clusters = 2		Initial number of clusters = 37	
Algorithm	μ-CPU Time (s)	σ-CPU Time (s)	μ-CPU Time (s)	σ-CPU Time (s)
CGA-I	0.5416	0.5533	2.1868	0.6676
CGA-II	0.0314	0.0276	0.1852	0.0567
CGA-III	0.0361	0.0264	0.2904	0.1796
CGA-IV	0.0361	0.0264	0.3234	0.1658

In both scenarios, the average CPU times of CGA-II were less than those for CGA-I. This result is supported by the Wilcoxon/Mann-Whitney test with $\alpha = 5\%$.

However, unlike the previous example ($k_{\text{initial}} \in \{2,\ldots,37\}$), the performances of both CGA-III and CGA-IV were not better than that of CGA-II. We believe that this result is due to the information loss caused by their centroid-based objective functions. This loss may make up for the computational savings resulting from using centroid-based objective functions, depending on the dataset and on the CGA initial population. Therefore, this subject deserves further investigations.

5 Conclusions and Future Works

This paper proposed modifications to a clustering genetic algorithm (CGA-I) aimed at automatically finding optimal clusterings in a given dataset. In essence, three incremental modifications to improve the CGA-I efficiency were proposed: (i) employing the K-Means Algorithm as a local search technique (CGA-II); (ii) adopting a simplified silhouette criterion based on centroids (CGA-III); (iii) using a more sensitive objective function (CGA-IV). Several simulations were performed to carefully evaluate the efficiency of every proposed modification. We have also performed statistical analyses on the results, which clearly indicated that CGA-II is more efficient than CGA-I. Some results (Table 1) also indicated that CGA-III and CGA-IV constitute promising strategies for the reduction of the computational time of the algorithm. For this reason, these algorithms will be further evaluated in real-world datasets. We are also going to investigate structural improvements to the CGAs genetic operators.

Acknowledgments

The authors thank both CNPq and FAPESP for their financial support.

References

1. Arabie, P., Hubert, L. J., An Overview of Combinatorial Data Analysis (Chapter 1). Clustering and Classification, ed. P. Arabie, L.J. Hubert, G. DeSoete, World Scientific, 1999.
2. Kaufman, L., Rousseeuw, P. J., Finding Groups in Data – An Introduction to Cluster Analysis, Wiley Series in Probability and Mathematical Statistics, 1990.
3. Liu, G.L., Introduction to combinatorial mathematics, McGraw Hill, 1968.
4. Park, Y., Song, M., A Genetic Algorithm for Clustering Problems, Proceedings of the Genetic Programming Conference, University of Wisconsin, July, 1998.
5. Hruschka, E. R., Ebecken, N.F.F., A genetic algorithm for cluster analysis, Intelligent Data Analysis (IDA), v.7, pp. 15-25, IOS Press, 2003.
6. Falkenauer, E., Genetic Algorithms and Grouping Problems, John Wiley & Sons, 1998.
7. Everitt, B.S., Landau, S., Leese, M., Cluster Analysis, Arnold Publishers, London, 2001.
8. Goldberg, D.E., Genetic Algorithms in Search, Optimization and Machine Learning, Addison Wesley Longmann, 1989.
9. Witten, I. H., Frank, E., Data Mining – Practical Machine Learning Tools and Techniques with Java Implementations, Morgan Kaufmann Publishers, USA, 2000.
10. Triola, M. F., Elementary Statistics, 7th Edition, Addison Wesley Longman, 1999.

The Hopfield Associative Memory Network: Improving Performance with the Kernel "Trick"

Cristina García and José Alí Moreno

Laboratorio de Computación Emergente,
Facultades de Ciencias e Ingeniería,
Universidad Central de Venezuela
{cgarcia, jose}@neurona.ciens.ucv.ve
http://neurona.ciens.ucv.ve/laboratorio

Abstract. This paper provides a new insight into the training of the Hopfield associative memory neural network by using the kernel theory drawn from the work on kernel learning machines and related algorithms. The kernel "trick" is used to define an embedding of memory patterns into (higher or infinite dimensional) memory feature vectors allowing the training of the network to be carried out in the feature space without ever explicitly representing it. The introduction of a kernel function in the training phase of the network considerably improves the performance of the network. This enhanced performance is experimentally investigated on sets of random memory patterns.

1 Introduction

Without doubt one of the most profusely studied neural network models has been the Hopfield associative memory neural net (HNN) [1, 2]. Indeed, from a superficial examination of the literature it becomes obvious that very much effort has been invested in its characterization. The interest has mainly been focused on the capacity of the network, on its learning rules and learning dynamics as well as on the dynamics of pattern retrieval [3–11]. The Hopfield neural model consists on a network of N two state $(S_i = \pm 1)$ neurons or processing units of the McCoulloch and Pitts type, fully connected to each other through synaptic weights w_{ij}. The neural network conforms a dynamical system that evolves in discrete time according to a relaxation dynamics. These neural models are commonly used as autoassociative content addressable memories. That is, they are able to retrieve a previously learnt memory pattern from an example which is similar to, or a noisy version of one of the memory patterns. A pattern is represented by associating each of its components to one of the processing units of the HNN. Memory recall proceeds by initially clamping the neurons to a starting pattern $\boldsymbol{S}(t = 0)$ and evolving in time the state of the network according to the dynamical rule towards the nearest attractor state. In consequence for the network to perform properly as an autoassociative memory, the activity pattern of the attractor state must correspond to a memory pattern. When this is the case it is said that $\boldsymbol{S}(t=0)$ lies in the basin of attraction of the corresponding memory pattern. Hence the purpose of a learning algorithm for the HNN is to find a synaptic weight matrix w_{ij} such that given p memory patterns, they become fixed points of the dynamics and that patterns resulting from small deviations of the memory patterns be in their basin of attraction.

The necessary condition for the memories to be fixed points of the dynamic is equivalent to stating that each processing unit of the network, acting as a linear binary classifier, solves a classification problem [5, 6]. The classification problem on each processing unit is defined by the set of memory patterns, acting as inputs with class labels equal to the associated memory component. In this respect most of the advanced learning rules proposed in the literature for the HNN [4, 5, 6] are sort of variants of the old perceptron learning rule for linear classifiers. On the other hand the condition that the basins of attraction be as large as possible is taken into account by training the processing units so as to maximize a stability operative criterion. The idea is that the output activity of the processing units be robust against small deviations of the input patterns from the memory patterns used in training.

The Adatron learning rule, an algorithm proposed in the statistical mechanics literature [6], implements this idea. From the point of view of linear classifiers the Adatron algorithm is an alternative formulation of the linear maximum margin classifier [10]. In summary the training of an optimal HNN is equivalent to the training of N linear maximum margin classifiers, one for each processing unit of the network. Recently [11] the Adatron algorithm has been generalized with the incorporation of the kernel trick. In this way the implementation simplicity of the Adatron is enhanced with the capability of working in a nonlinear feature space as support vector machines do. The result is a kernel machine that maximizes the margin in a feature space, which is equivalent to nonlinear decision boundaries in input space. The basic idea behind the "kernel trick" is that under certain conditions the kernel function can be interpreted as an inner product in a high dimensional Hilbert space (feature space). A fact that has been used extensively in generating non-linear versions of conventional linear supervised and unsupervised learning algorithms [10].

In the present case, the application of this procedure to the HNN defines an embedding of memory patterns into (higher or infinite dimensional) memory feature vectors allowing the training of the network to be carried out in the feature space without ever explicitly representing it. We refer to this enhanced neural model a kernel-HNN. The result is that the performance of the network is improved in two aspects. First, the condition that arbitrary memory patterns be attractors of the network dynamics is readily met by the selection of an adequate kernel. That is the classification problem at each processing unit has always a solution. Second, the basin of attraction of the memory patterns are increased improving the capacity of recall of the network. Experimental simulation runs on a kernel-HNN trained with a set of uncorrelated random memory patterns for different memory loads and several kernel functions are carried out. The experimental results show a significant performance gain in the pattern retrieval dynamics with respect to the linear HNN model. Qualitative measures of the increase of the basins of attraction are reported. The organization of the paper is as follows: in section 2 an introduction to the the formalism of the HNN model is presented; in section 3, the Kernel extension of the Adatron learning algorithm is described and in section 4 the kernel-HNN model introduced; in section 5 the experiments and results are discussed and, in section 6, the conclusions are presented.

2 The Hopfield Associative Memory Network

The HNN consists on a network of N McCoulloch and Pitts type processing units fully connected to each other through a matrix of synaptic weights w_{ij}, with no self coupling $(w_{ii} = 0)$. The exclusion of the self coupling terms enlarge the basins of attraction [8]. The dynamics of such a system is taken to be a simple zero-temperature Monte Carlo process

$$S_i(t+1) = sign(\sum_j w_{ij} S_j(t)) = sign\left(\langle \boldsymbol{w_i} \cdot \boldsymbol{S}(t)\rangle\right) \tag{1}$$

where $\boldsymbol{S}(t)$ is the vector of activities of the network at time t $(S_i = \pm 1)$, $\boldsymbol{w_i}$ the weight vector of the ith processing unit and the brackets denotes dot product. The update of the network is performed asynchronously. The configuration of synaptic weights is generally adjusted by a learning algorithm such that, the given p memory patterns

$$\boldsymbol{Y}^\nu = (y_1^\nu, \ldots, y_N^\nu)^T, \quad y_i^\nu = \pm 1, \quad \nu = 1, \ldots, p \tag{2}$$

become locally stable fix points of the dynamics. That is, the set of inequalities

$$y_i^\nu h_i^\nu = y_i^\nu \sum_j w_{ij}\; y_j^\nu \geq c > 0 \tag{3}$$

must be satisfied for every memory pattern ν and processing unit i. Since the diagonal elements of the weight matrix are null, the above set of inequalities decouple and each processing unit can be treated independently. We can then consider the case for one processing unit only, say $i = 1$, and omit the index i in the following. The condition for the stability of the memories is then written

$$y^\nu h^\nu = y^\nu \sum_j w_j\; y_j^\nu = \gamma^\nu \geq c > 0 \tag{4}$$

where w_j are the components of the weight vector of a linear processing unit with component $w_i = 0$ (i is the index of the processing unit). This condition denotes the positiveness of the functional margins γ^ν, of the input vectors $\boldsymbol{Y}^\nu$ with labels y^ν (ith component of the memory pattern) with respect to the linear classifier defined by the weight vector $\boldsymbol{w}$. This is the separability condition of the set of memory patterns and corresponding labels by the linear classifier on the *i*th processing unit [10]. The associative memory network will be properly trained when all classification problems are simultaneously separable. In another case the recall dynamics will be unstable and converge to a pattern of activities different from a memory.

Fulfillment of inequality (4) is a necessary condition for the HNN to operate as an associative memory but in addition large basins of attraction are desirable for each memory pattern. Thus the fix points should be stable against many component inversions of the activity patterns of the network in its time evolution. As a sufficient condition it has been argued [5, 6] that the functional margins γ^ν in (4) should be large relative to the magnitudes of the synaptic weights. A normalized measure of stability for the processing units is then defined as:

$$\hat{\gamma} = \min_{\nu}(\hat{\gamma}^\nu) \quad \hat{\gamma}^\nu = \frac{\gamma^\nu}{||\boldsymbol{w}||} = \frac{y^\nu \sum_j w_j \; y_j^\nu}{||\boldsymbol{w}||} \tag{5}$$

expression which is equivalent to the geometrical margin [10] of the set of input memories with respect to the linear classifier defined by the weight vector $\boldsymbol{w}$.

The Adatron algorithm is a learning algorithm which attains optimal stability, i.e. a configuration of weights with maximum geometrical margin for a set of arbitrary memory patterns. It is an alternative formulation of the maximum margin classifier algorithm well known in the machine learning literature. An enhanced version of the algorithm with the application of the kernel trick has been presented [11]. In this way the learning proceeds in a high-dimensional memory feature space providing a non-linear version of the conventional linear perceptronlike learning algorithm. By introducing Kernels into the algorithm the measure of stability of the processing units is maximized in memory feature space, that is the stability conditions for the memories, inequalities in (4), become nonlinear in input memory space. This allows that, with a suitable kernel choice, sets of memory patterns for which the conditions (4) could not be fulfilled in input memory space, non separable set, do satisfy them in memory feature space. Furthermore as we experimentally show the basin of attraction of the memories are increased and the information content of the network is enhanced.

3 The Kernel Trick in the Adatron Algorithm

The basic idea behind the Adatron algorithm of Anlauf and Biehl [6] is to induce a configuration of synaptic weights that maximize the stability of the processing unit (5). They assume a scale for the weight vectors where the positive constant c in relation (4) is unity. In this way the algorithm is derived from the problem of minimizing the norm of the weight vector subject to the restrictions that the functional margins of the memory patterns be greater or equal to unity:

$$minimize \; \frac{1}{2}||\boldsymbol{w}||^2 \quad s.t. \; \gamma^\nu = y^\nu \sum_j w_j \; y_j^\nu \geq 1 \; \forall \nu = 1, \ldots, p \tag{6}$$

In the dual formulation of this optimization problem they show that the solution has the form:

$$\boldsymbol{w} = \frac{1}{N} \sum_{\nu=1}^{p} \alpha_\nu \, y^\nu \boldsymbol{Y}^\nu \tag{7}$$

where the α_ν are positive Lagrange multipliers which can be interpreted as a measure of the information contribution that each memory pattern does to the synaptic weights. These dual variables are learned, from a *tabula rasa* initialization, by an iterative procedure where the memory patterns are presented repeatedly and the following corrections are applied for each memory:

$$\delta\alpha_\nu = \max\{-\alpha_\nu, \lambda(1-\gamma^\nu)\} \tag{8}$$

where λ is a learning rate. The learning is done when the minimum functional margin equals unity. This algorithm is presented with theoretical guaranties of convergence to the optimal solution and of a rate of convergence exponentially fast in the number of iterations, provided that a solution exists.

It is interesting to note that the above optimization problem can be entirely formulated in terms of dot products hence it can be solved in an arbitrary feature feature space induced by a kernel function. A symmetric function $K(\boldsymbol{x}, \boldsymbol{y})$ is a kernel if it fulfills Mercer's condition, i.e. the function K is (semi) positive definite. When this is the case there exists a mapping ϕ such that it is possible to write $K(\boldsymbol{x}, \boldsymbol{y}) = \langle \phi(\boldsymbol{x}) \cdot \phi(\boldsymbol{y}) \rangle$. That is the kernel represents a dot product on a different space F called feature space into which the original vectors are mapped. With the introduction of a suitable kernel function the above learning procedure of the Adatron can be carried out in an arbitrary feature space where on the one hand the linear separability of the problem be guaranteed for every processing unit and on the other, the high dimensionality of the feature space produces an increment on the capacity of the network.

Summarizing, in [11] is shown that the learning rule given in expression (8) remains invariant when considering learning in feature space and the form of the resulting kernel classifier on each processing unit of the network is

$$S_i(t+1) = sign\left(\sum_{\nu=1}^{p} \alpha_\nu \, y_i^\nu K(\boldsymbol{Y}^\nu, \boldsymbol{S}(t)) \right) \tag{9}$$

$K(\boldsymbol{Y}^\nu, \boldsymbol{S})$ is the kernel function and the α_ν are the Lagrange multipliers resulting from the solution of the optimization problem (6) in its dual representation. They are obtained in the training procedure. In the literature [10] the most common used kernels are the linear, polynomial, and Gaussian kernels shown in relation (10). The linear kernel is the simple dot product in input space whereas the other kernel functions represent dot products in arbitrary feature space.

$$\begin{aligned} K_l(\boldsymbol{Y}, \boldsymbol{S}) &= \langle \boldsymbol{Y} \cdot \boldsymbol{S} \rangle \\ K_p(\boldsymbol{Y}, \boldsymbol{S}) &= (\langle \boldsymbol{Y} \cdot \boldsymbol{S} \rangle + b)^n \\ K_G(\boldsymbol{Y}, \boldsymbol{S}) &= \exp(-\|\boldsymbol{Y} - \boldsymbol{S}\|^2/\sigma^2) \end{aligned} \tag{10}$$

In this work, experiments with the three of the kernel functions in (10) are carried out.

4 Kernel Hopfield Network

Expression (9) is a generalization of the dynamic equation in (1) defining the dual representation of the dynamic of the HNN in feature space:

$$S_i(t+1) = sign\left(\langle \boldsymbol{w_i} \cdot \boldsymbol{S}(t)\rangle\right) = sign\left(\sum_{\nu=1}^{p} \alpha_\nu \; y_i^\nu K(\boldsymbol{Y}^\nu, \boldsymbol{S}(t))\right) \tag{11}$$

It is important to note that the described formalism does not provide an equation for the synaptic weights similar to relation (7), as a linear combination of the transforms of the memory patterns in feature space. The reason for this is simply that the explicit mapping of the input memories into feature space is not known. Nevertheless rewriting the definitions (10) of the kernel functions in terms of the products of components of the involved vectors in input space an expression equivalent to (7) for the weight vectors can be given as a generalized product of functions of the memory components. It is shown in [10] that for the polynomial kernel this is readily done. As an example let us consider a simple polynomial kernel with $b = 0$ and $n = 2$. Equation (11) can be written

$$S_i(t+1) = sign\left(\langle \boldsymbol{w_i} \cdot \boldsymbol{S}(t)\rangle\right) = sign\left(\sum_{\nu=1}^{p} \alpha_\nu \; y_i^\nu \langle \boldsymbol{Y}^\nu \cdot \boldsymbol{S}(t)\rangle^2\right) \tag{12}$$

since

$$\langle \boldsymbol{Y}^\nu \cdot \boldsymbol{S}(t)\rangle^2 = \sum_{k=1}^{N} \sum_{j=1}^{N} y_k^\nu \, y_j^\nu \, S_k \, S_j \tag{13}$$

Inserting (13) in (12)

$$S_i(t+1) = sign\left(\sum_{(k,j)=(1,1)}^{(N,N)} S_k \, S_j \sum_{\nu=1}^{p} y_i^\nu \, \alpha_\nu \, y_k^\nu \, y_j^\nu\right) \tag{14}$$

The argument of the function *sign* in expression (14) is a generalized inner product of feature vectors with components $S_k S_j$ and $y_k^\nu y_j^\nu$. It can be recognized that the summatory on the memory patterns $y_k^\nu y_j^\nu$ is of the same form as expression (7). The writing of an equivalent expression for the Gaussian kernel is a more involved procedure that takes us out of the scope of this paper, but in general one such expression can be written.

In conclusion the kernel trick allows a straightforward generalization of the Hopfield network to a to higher dimensional feature space. The important advantage of this procedure is that in principle all processing units can be trained to optimal stability. Such a network of fully optimal processing units show several important improvements that can be experimentally shown: larger memory capacity, increased memory recall capacity and bigger basins of attraction.

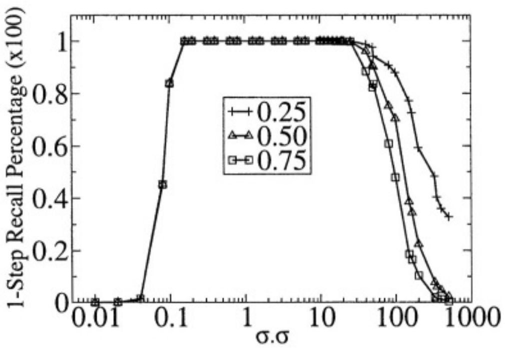


Fig. 1. Fraction of correctly recognized patterns in one dynamical step as a function of the parameter of the Gaussian kernel, for three values of the memory loading parameter: 0.25, 0.5 and 0.75

5 Experiments and Results

In order to investigate the kernel-HNN in a systematic fashion a statistical analysis of several simulations were carried out. Networks were trained for different memory loads. For every trained network, experiments on the recovery of each memory pattern contaminated with a given percentage of random noise were repeated 50 times. The reported results are the values of the fraction of correctly recovered patterns. A $N = 100$ processing unit Hopfield network was used throughout. Three kinds of kernel functions were investigated: linear, polynomial with $b = 0.1$ and $n = 2$; and Gaussian kernel. The parameter value of the Gaussian was established experimentally. The dynamic of the network is synchronous in the one step memory recall experiment and asynchronous in the others.

5.1 One Step Memory Recall

In these experiments the fraction of correctly recovered memory patterns, in one dynamical step, are measured as a function of the kernel parameter for various values of the memory loading parameter $\alpha = p/N$ $(0.25, 0.5, 0.75)$. The experiments measure a sort of direct basin of attraction of the memories. It was carried out only on the network with Gaussian kernel in order to establish an adequate value of the kernel parameter. The kernel parameter is swept over several orders of magnitude. The results are shown in Figure 1.

It can be appreciated that the direct basins of attraction suffer a slight decrease with memory loading but remain essentially invariant over a broad range of the kernel parameter values. Since greater parameter values produce sparser dual representations, the results suggest that a reasonable good choice for the value of this parameter is $\sigma^2 = 25$. This value is used in the rest of the experiments.

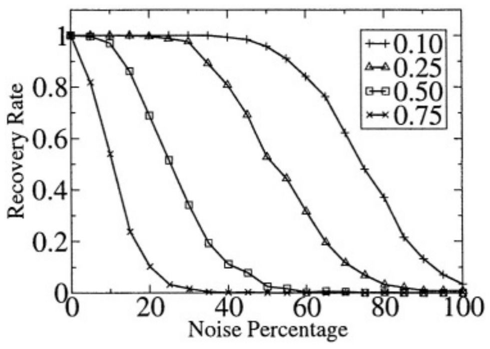


Fig. 2. Fraction of correctly recognized patterns in a linear kernel network as a function of the initial random noise contamination, for four values of the memory loading parameter: 0.10, 0.25, 0.5 and 0.75

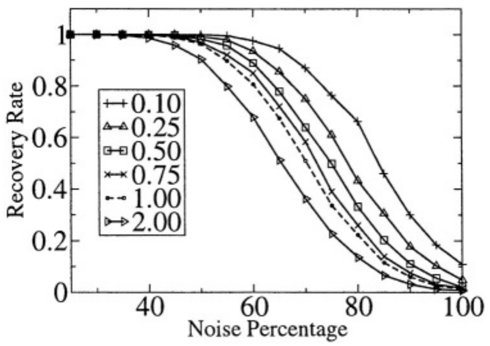


Fig. 3. Fraction of correctly recognized patterns in a polynomial kernel network as a function of the initial random noise contamination, for six values of the memory loading parameter: 0.10, 0.25, 0.5, 0.75, 1.0 and 2.0

5.2 Experiments on the Capacity of Memory Recall

In these experiments the fraction of correctly recovered memory patterns are measured as a function of the magnitude of the initial distortion of the memories (noise contamination) for various values of the loading parameter. The experiments were carried out for the three kernel types. In figure 2 the results for the linear kernel are shown.

It can be appreciated that the basins of attraction shrink with increasing value of the loading parameter. Already for loading parameter 0.75 the network does not support any contamination of the input stimulus. This is the typical behavior of standard Hopfield Networks [8]. In Figure 3 the results for a network with polynomial kernel are presented.

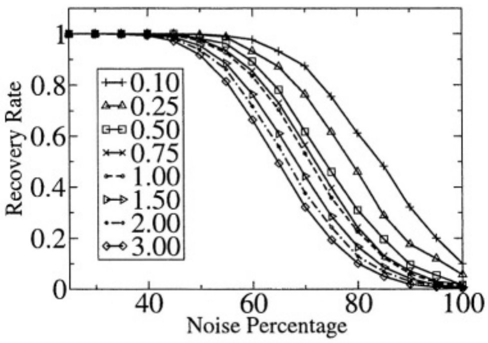


Fig. 4. Fraction of correctly recognized patterns in a Gaussian kernel network, $\sigma^2 = 25$ as a function of the initial random noise contamination, for eight values of the memory loading parameter: 0.10, 0.25, 0.5, 0.75, 1.0, 1.5, 2.0 and 3.0

In this case the basins of attraction do decrease in a moderate way with increasing values of memory loading but remain robust for loading parameter values as high as 2. The network is tolerant to input contaminations as high as 40% for the highest memory load of 2.0. This is indicative of the presence of large basins of attraction. Finally in Figure 4 the results for a network with Gaussian kernel are shown.

It can be appreciated that the basins of attraction again with this kernel decrease in a moderate way with increasing values of memory loading but remain robust for loading parameter values as high as 3. The network again evidence the presence of large basins of attraction.

It is clear from this series of experiments that the kernel-HNN model is superior to the standard linear model in several aspects: increased memory capacity, larger basins of attraction and hence a great robustness to noise contamination of the stimuli patterns.

6 Conclusions

We have presented a generalization of the Hopfield auto-associative memory network taking advantage of the kernel trick already well known in the Machine Learning literature. It is experimentally shown that the kernel-Hopfield networks are clearly superior in performance to the standard linear model. Important observables of these models like memory capacity, recall capacity, noise tolerance and size of the basins of attraction are clearly improved by the introduction of the kernel formalism. That is, an implicit mapping of the network to a higher dimensional feature space. In this generalization procedure the learning algorithm of the network remains in its simple form as it is applied to linear networks. This means that the training complexity is by no means affected by the generalization. Nevertheless, the training of an N processing units kernel-HNN model implies the calculation of N kernel-Adatron maximal margin classifiers over the set of memory patterns. Resulting in a considerable computational burden as N increases one or two orders of magnitude. It is clear that further work, both theoretical as computational is necessary in the model. Computationally is necessary to investigate the possibility of implementing larger networks (1000 – 10000 processing units) in order to make the model attractive for real life applications. We are engaged in the parallel implementation of these models on clusters of workstations. Finally more theoretical work is also needed in the formalization of this generalized neural network model.

References

1. Muller, B., Reinhardt, J.: Neural Networks. An Introduction. Springer - Verlag, Berlin (1990)
2. Hertz, J., Krogh, A., Palmer, R.: Introduction to the Theory of Neural Computation. Addison-Wesley, Redwood City, CA (1991)
3. Personnaz, L., Guyon, I., Dreyfus, J.: Collective computational properties of neural networks: New learning mechanisms. J. Physique Lett. **16** (1985) 1359
4. Diederich, S., Opper, M.: Learning of correlated patterns in spin-glass networks by local learning rules. Phys. Rev. Lett. **58** (1987) 949
5. Krauth, W., Mezard, M.: Learning algorithms with optimal stability in neural networks. J. Phys. A **20** (1987) 1745
6. Anlauf, J., Biehl, M.: The adatron - an adaptive perceptron algorithm. Europhysics Letters **10** (1989) 687–692
7. Opper, M.: Learning times of neural networks: Exact solution for a perceptron algorithm. Phys. Rev. A **38** (1988) 3824
8. Gardner, E.: The space of interactions in neural network models. J. Phys. A **21** (1988) 257
9. Gardner, E., Derrida, B.: Optimal storage properties of neural network models. J. Phys. A **21** (1988) 271
10. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines. Cambridge University Press (2000)
11. Friest, T., Campbell, C., Cristianini, N.: The kernel-adatron: A fast and simple learning procedure for support vector machines. In: Proceedings of the Fifteenth International Conference on Machine Learning. Morgan - Kaufmann, San Francisco, CA (1998)

A Cultural Algorithm with Differential Evolution to Solve Constrained Optimization Problems

Ricardo Landa Becerra and Carlos A. Coello Coello

CINVESTAV-IPN (Evolutionary Computation Group),
Dpto. de Ing. Elect./Secc. Computación,
Av. IPN No. 2508, Col. San Pedro Zacatenco,
México, D.F. 07300, Mexico
rlanda@computacion.cs.cinvestav.mx
ccoello@cs.cinvestav.mx

Abstract. A cultural algorithm is proposed in this paper. The main novel feature of this approach is the use of differential evolution as a population space. Differential evolution has been found to be very effective when dealing with real valued optimization problems. The knowledge sources contained in the belief space of the cultural algorithm are specifically designed according to the differential evolution population. Furthermore, we introduce an influence function that selects the source of knowledge to apply the evolutionary operators. Such influence function considerably improves the performance when compared to a previous version of the algorithm (developed by the same authors). We use a well-known set of test functions to validate the approach, and compare the results with respect to the best constraint-handling technique known to date in evolutionary optimization.

1 Introduction

Cultural algorithms are techniques that add domain knowledge to evolutionary computation methods. They are based on the assumption that domain knowledge can be extracted during the evolutionary process, by means of the evaluation of each point generated [11]. This process of extraction and use of the information, has been shown to be very effective in decreasing computational cost while approximating global optima, in unconstrained, constrained and dynamic optimization [13, 3, 7, 15]. Cultural algorithms are made of two main components: the population space, and the belief space [12]. The **population space** consists of a set of possible solutions to the problem, and can be modeled using any population based technique, e.g. genetic algorithms [5]. The **belief space** is the information repository in which the individuals can store their experiences for the other individuals to learn them indirectly. In cultural algorithms, the information acquired by an individual can be shared with the entire population. Both spaces (i.e., population space and belief space) are linked through a communication protocol, which states the rules about the individuals that can contribute to the belief space with its experiences (the acceptance function), and the way the belief space can influence to the new individuals (the influence function). Those interactions are depicted in Figure 1.

Originally, when cultural algorithms were applied to real parameter optimization, genetic algorithms were used as a population space [11]. Later on, evolutionary pro-

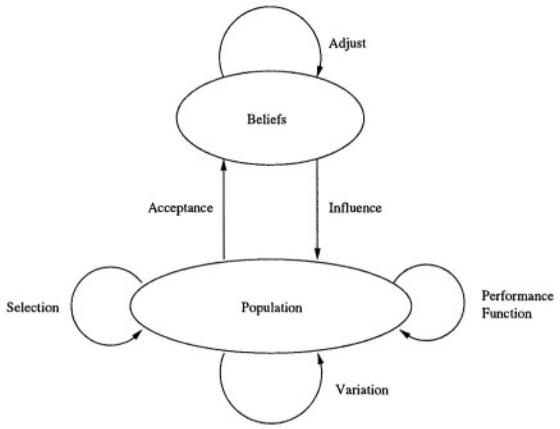


Fig. 1. Spaces of a cultural algorithm

gramming appeared as a better choice [2] for the population space than genetic algorithms when dealing with constrained search spaces [4, 7, 3]. Recently, particle swarm [8] has also been proposed as a population space [6], turning the direction to use new evolutionary methods with better performance in real parameter optimization.

Differential evolution [10] is a recently developed evolutionary algorithm, focused on solving real paramenter optimization problems. Differential evolution has been found to be a very robust optimization technique [16]. However, to the authors' best knowledge, we are the first to propose the use of differential evolution as the population space of a cultural algorithm [9]. This paper precisely presents an extension of our previous work in which we did a preliminary exploration of the potential of differential evolution to be "culturized" [9].

2 Previous Work

Reynolds et al. [13] and Chung & Reynolds [3] have explored the use of cultural algorithms for global optimization with very encouraging results. Chung and Reynolds use a hybrid of evolutionary programming and GENOCOP in which they incorporate an interval constraint-network to represent the constraints of the problem at hand. In more recent work, Jin and Reynolds [7] proposed an n-**dimensional** regional-based schema. The idea of Jin and Reynolds' approach is to build a map of the search space which is used to derive rules about how to guide the search of the evolutionary algorithm (avoiding infeasible regions and promoting the exploration of feasible regions). Using the same population space (evolutionary programming), Saleem proposes a cultural algorithm for dealing with dynamic environments [15]. Saleem adds history and domain knowledge to Chung's situational and normative knowledge, and Jin's topographical knowledge. In [6], Iacoban et al. change the evolutionary programming algorithm from the population space for a particle swarm optimizer [8]. They make an analysis of the effects of the belief space over the evolutionary process, showing the similarities with the approach in which evolutionary programming is adopted.

3 Differential Evolution

Differential evolution is an evolutionary algorithm proposed by Price and Storn [10], whose main design emphasis is real parameter optimization. Differential evolution is based on a mutation operator, which adds an amount obtained by the difference of two randomly chosen individuals of the current population. The basic algorithm of differential evolution is shown in Figure 2, where the problem to be solved has n decision variables, *F* and *CR* are parameters given by the user, and $x_{i,j}$ is the i-th decision variable of the j-th individual in the population. The authors of the differential evolution algorithm have suggested that by computing the difference between two individuals randomly chosen from the population, the algorithm is actually estimating the gradient in that zone (rather than in a point). This approach is also rather efficient way to self-adapt the mutation operator. The version of differential evolution shown in Figure 2, is called DE/rand/1/bin, and is recommended to be the first choice when trying to apply differential evolution [10]. That is the reason why we adopted it for the work reported in this paper.

```
Generate initial population of size $popsize$
Do
    For each individual $j$ in the population
        Generate three random integers, $r_1, r_2$ and $r_3 \in (1, popsize)$, with $r_1 \neq r_2 \neq r_3 \neq j$
        Generate a random integer $i_{rand} \in (1, n)$
        For each parameter $i$
            $x'_{i,j} = \begin{cases} x_{i,r3} + F * (x_{i,r1} - x_{i,r2}) & \text{if } rand(0,1) < CR \text{ or } i = i_{rand} \\ x_{i,j} & \text{otherwise} \end{cases}$
        End For
        Replace $x_j$ with the child $x'_j$, if $x'_j$ is better
    End For
Until the termination condition is achieved
```

Fig. 2. Pseudo-code of the differential evolution algorithm adopted in this work (this version is called DE/rand/1/bin)

4 Our Proposed Approach

The proposed approach uses differential evolution in the population space. A pseudo-code of the cultured differential evolution is shown in Figure 3.

In the initial steps of the algorithm, a population of *popsize* individuals is created, as well as a belief space. For the children generation, the variation operator of the differential evolution algorithm is influenced by the belief space. Since we want to solve constrained optimization problems, the fitness function by itself does not provide enough information as to guide the search properly. To determine if a child is better than its parent, and it can replace it, we use the following rules:

1. A feasible individual is always better than an infeasible one.
2. If both are feasible, the individual with the best objective function value is better.

```
Generate initial population
Evaluate initial population
Initialize the belief space
Do
    For each individual in the population
        Apply the variation operator influenced by a randomly chosen knowledge source
        Evaluate the child generated
        Replace the individual with the child, if the child is better
    End for
    Update the belief space with the accepted individuals
Until the termination condition is achieved
```

Fig. 3. Pseudo-code of the cultured differential evolution

3. If both are infeasible, the individual with less amount of constraint violations is better, measuring violations with normalized constraints.

4.1 The Belief Space

In our approach, the belief space is divided in four knowledge sources, described next.

Situational Knowledge. Situational knowledge consists of the best exemplar *E* found along the evolutionary process. It represents a leader for the other individuals to follow. The variation operators of differential evolution are influenced in the following way:

$$x'_{i,j} = E_i + F * (x_{i,r1} - x_{i,r2})$$

where E_i is the i-th component of the individual stored in the situational knowledge. This way, we use the leader instead a randomly chosen individual for the recombination, getting the children closer to the best point found. The update of the situational knowledge is done by replacing the stored individual, *E,* by the best individual found in the current population, x_{best}, only if x_{best} is better than *E.*

Normative Knowledge. The normative knowledge contains the intervals for the decision variables where good solutions have been found, in order to move new solutions towards those intervals. Thus, the normative knowledge has the structure shown in Figure 4.

l_1	u_1	l_2	u_2	$\cdots$	l_n	u_n
L_1	U_1	L_2	U_2	$\cdots$	L_n	U_n

dm_1	dm_2	$\ldots$	dm_n

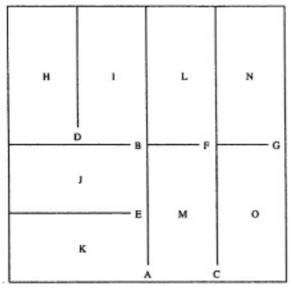
Fig. 4. Structure of the normative knowledge

In Figure 4, l_i and u_i are the lower and upper bounds, respectively, for the i-th decision variable, and L_i and U_i are the values of the fitness function associated with that bound. Also, the normative knowledge includes a scaling factor, dm_i, toinfluence the mutation operator adopted in differential evolution. The following expression shows the influence of the normative knowledge on the variation operators:

$$x'_{i,j} = \begin{cases} x_{i,r3} + F * |x_{i,r1} - x_{i,r2}| & \text{if } x_{i,r3} < l_i \\ x_{i,r3} - F * |x_{i,r1} - x_{i,r2}| & \text{if } x_{i,r3} > u_i \\ x_{i,r3} + \frac{u_i - l_i}{dm_i} * F * (x_{i,r1} - x_{i,r2}) & \text{otherwise} \end{cases}$$

The update of the normative knowledge can reduce or expand the intervals stored on it. An expansion takes place when the accepted individuals do not fit in the current interval, while a reduction occurs when all the accepted individuals lie inside the current interval, and the extreme values have a better fitness and are feasible. The values dm_i are updated with the greatest difference $|x_{i,r1} - x_{i,r2}|$ found during application of the variation operators of the previous generation.

Topographical Knowledge. The usefulness of the topographical knowledge is to create a map of the fitness landscape of the problem during the evolutionary process. It consists of a set of cells, and the best individual found on each cell. The topographical knowledge, also, has an ordered list of the best b cells, based on the fitness value of the best individual on each of them. For the sake of a more efficient memory management, in the presence of high dimensionality (i.e., too many decision variables), we use an spatial data structure, called k-d tree, or k-dimensional binary tree [1]. In k-d tree, each node can only have two children (or none, if it is a leaf node), and represents a division in half for any of the k dimensions (see Figure 5).



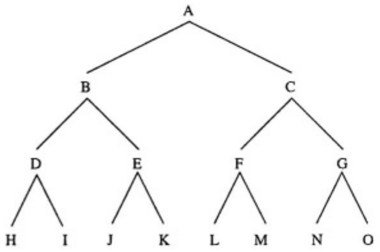


Fig. 5. Example of the partition of a two dimensional space by a k-d tree

The influence function tries to move the children to any of the b cells in the list:

$$x'_{i,j} = \begin{cases} x_{i,r3} + F * |x_{i,r1} - x_{i,r2}| & \text{if } x_{i,r3} < l_{i,c} \\ x_{i,r3} - F * |x_{i,r1} - x_{i,r2}| & \text{if } x_{i,r3} > u_{i,c} \\ x_{i,r3} + F * (x_{i,r1} - x_{i,r2}) & \text{otherwise} \end{cases}$$

where $l_{i,c}$ and $u_{i,c}$ are the lower and upper bounds of the cell c, randomly chosen from the list of the b best cells. The update function splits a node if a better solution is found in that cell, and if the tree has not reached its maximum depth. The dimension in which the division is done, is the one that has a greater difference between the solution stored and the new reference solution (i.e., the new solution considered as the "best" found so far).

History Knowledge. This knowledge source was originally proposed for dynamic objective functions, and it was used to find patterns in the environmental changes. History knowledge records in a list, the location of the best individual found before each environmental change. That list has a maximum size w. The structure of history knowledge is shown in Figure 6, where e_i is the best individual found before the i-th environmental change, ds_i is the average distance of the changes for parameter i, and dr_i is the average direction if there are changes for parameter i. In our approach, instead of detecting changes of the environment, we store a solution if it remains as the best one during the last p generations. If this happens, we assume that we are trapped in a local optimum.

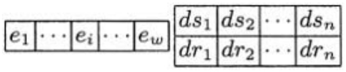


Fig. 6. Structure of the history knowledge

The expression of the influence function of the history knowledge is the following:

$$x'_{i,j} = \begin{cases} x_{i,e_w} + dr_i * F * |x_{i,r1} - x_{i,r2}| & \text{if } rand(0,1) < \alpha \\ x_{i,e_w} + \frac{1.5 * ds_i}{dm_i} * (x_{i,r1} - x_{i,r2}) & \text{if } rand(0,1) < \beta \\ rand(lb_i, ub_i) & \text{otherwise} \end{cases}$$

where x_{i,e_w} is the i-th decision variable of the previous best e_w stored in the list of the history knowledge, dm_i is the maximum difference for the i-th variable, stored in the normative knowledge, lb_i and ub_i are the lower and upper bounds of the variable x_i, given as input for the problem, and $rand(a, b)$ is a random number between a and b. To update the history knowledge, we add to the list any local optima found during the evolutionary process. If the list has reached its maximum length w, the oldest element is discarded. The average distances and directions of change are calculated by:

$$ds_i = \frac{\sum_{k=1}^{w-1} |x_{i,e_{k+1}} - x_{i,e_k}|}{w-1}$$

$$dr_i = sgn\left(\sum_{k=1}^{w-1} sgn\left(x_{i,e_{k+1}} - x_{i,e_k}\right)\right)$$

where the function $sgn(a)$ returns the sign of a.

4.2 Acceptance Function

The number of individuals accepted for the update of the belief space is computed according the design of a dynamic acceptance function by Saleem [15]. The number of accepted individuals decreases while the number of generation increases. Saleem [15] suggests to reset the number of accepted individuals when an environmental change occurs. In our case, we reset the number of accepted individuals when the best solution has not changed in the last p generations. We get the number of accepted individuals, $n_{accepted}$, with the following expression:

$$n_{accepted} = \left\lfloor \%p * popsize + \frac{(1 - \%p) * popsize}{g} \right\rfloor$$

where $\%p$ is a parameter given by the user, in (0, 1]; Saleem [15] suggests using 0.2. g is the generation counter, but is reset to 1 when the best solution has no changed in the last p generations.

4.3 Main Influence Function

The main influence function is responsible for choosing the knowledge source to be applied to the variation operator of differential evolution. At the beginning, all the knowledge sources have the same probability to be applied, $\%p_{ks} = \frac{1}{4}$, because there are 4 knowledge sources; but during the evolutionary process, the probability of the knowledge source *ks* to be applied is:

$$\%p_{ks} = 0.1 + 0.6\frac{v_{ks}}{v}$$

where v_{ks} are the times that an individual generated by the knowledge source *ks* outperforms its parent in the current generation, and v are the times that an individual generated (by any knowledge source) outperforms its parent in the current generation. The lower bound of the value $\%p$ is 0.1, to ensure that any knowledge source has always a probability > 0 to be applied. If $v = 0$ during a generation, $\%p_{ks} = \frac{1}{4}$, as in the beginning. This main influence function is the most important modification with respect to the previous version of this algorithm ([9]).

5 Comparison of Results

To validate our approach, we adopted the well-known benchmark included in [14] which has been often used in the literature to validate new constraint-handling techniques. For a full description of the test functions adopted, the reader should refer to [14]. The parameters used by our approach are the following: *popsize* = 100, maximum number of generations = 1000, the factors of differential evolution are F = 0.5 and CR — 1, maximum depth of the k-d tree = 12 length of the best cells list $b = 10$, the size of the list in the history knowledge $w = 5$, $\alpha = \beta = 0.45$, and $\%p = 0.2$. The values shown in tables were obtained executing 30 independent runs per problem.

Table 1 shows the results obtained by our approach. The results obtained by the stochastic ranking method (the best constraint-handling technique proposed for evolutionary algorithms known to date) are shown in Table 3. Our results are also compared to a previous version of our algorithm [9] in Table 2. The results of Runarsson and Yao were obtained with 350,000 evaluations of the fitness function. Our approach required only 100,100 evaluations (in both versions, the previous [9] and the one reported in this paper).

As can be seen in Tables 1 and 2, the new version of our algorithm exhibits a better performance than our previous version in all cases, except in g05, where its variablility

Table 1. Results obtained by our cultured differential evolution approach

TF	Optimal	Best	Mean	Worst	Std Dev
g01	**-15**	-14.999863	-14.999351	-14.998283	0.000333
g02	**0.803619**	0.793829	0.735590	0.620843	0.049941
g03	**1**	**1.000000**	0.896800	0.69272	0.080994
g04	**-30665.539**	**-30665.538672**	**-30665.538672**	**-30665.538672**	0.000000
g05	**5126.4981**	5126.558552	5198.202774	5323.865946	59.633275
g06	**-6961.8138**	**-6961.813876**	**-6961.813876**	**-6961.813876**	0.000000
g07	**24.3062091**	24.575518	24.575520	24.575526	0.000002
g08	**0.095825**	**0.095825**	**0.095825**	**0.095825**	0.000000
g09	**680.6300573**	**680.630057**	**680.630057**	**680.630057**	0.000000
g10	**7049.25**	**7049.248134**	7049.248489	7049.249942	0.000362
g11	**0.75**	**0.750000**	0.777469	0.898055	0.044560
g12	**1**	**1.000000**	**1.000000**	**1.000000**	0.000000
g13	**0.0539498**	0.056180	0.288324	0.39210	0.161230

Table 2. Results obtained by our previous version of cultured differential evolution [9]

TF	Optimal	Best	Mean	Worst	Std Dev
g01	**-15**	14.996953	13.214513	5.999896	2.985388
g02	**0.803619**	0.616900	0.517901	0.419959	0.066237
g03	**1**	**1.000000**	0.821397	0.600900	0.144609
g04	**-30665.539**	**-30665.539177**	-30665.538824	-30665.538672	0.000244
g05	**5126.4981**	5126.563220	5136.862081	5184.827897	19.569988
g06	**-6961.8138**	**-6961.813876**	**-6961.813876**	**-6961.813876**	0.000000
g07	**24.3062091**	24.575671	24.585679	24.650253	0.023298
g08	**0.095825**	**0.095825**	**0.095825**	**0.095825**	0.000000
g09	**680.6300573**	**680.630057**	**680.630057**	**680.630057**	0.000000
g10	**7049.25**	**7049.251189**	7049.284777	7049.372205	0.040707
g11	**0.75**	0.757500	0.779440	0.854357	0.039593
g12	**1**	**1.000000**	**1.000000**	**1.000000**	0.000000
g13	**0.0539498**	0.054903	0.314341	0.426815	0.181099

is higher. The current version now reaches the optimum in g11, and also consistently reaches the optimum of g04. When compared to stochastic ranking (see Table 3), our cultured differential evolution algorithm turns out to be very competitive. Our approach reached the global optimum in eight problems, and stochastic ranking did it in nine. However, with the exception of g02 and g13 (where stochastic ranking was a clear winner), in all the other problems the results obtained by our approach are very close to the global optimum. An additional aspect that we found quite interesting is that our approach presented in most cases a low standard deviation, improving on the robustness of stochastic ranking in several cases. A remarkable example is g10, where stochastic ranking was not able to reach the global optimum and presented a high variability of results. Another example is g06, where stochastic ranking also presented a higher variability than our approach. In contrast, stochastic ranking showed a more robust behavior than our approach in g01, g03, g05 and g11.

6 Conclusions and Future Work

In this paper we introduce a cultural algorithm, which uses differential evolution. This work improves on our previous attempt to develop a cultural algorithm that uses a

Table 3. Results reported for stochastic ranking [14]

TF	Optimal	Best	Mean	Worst	Std Dev
g01	**-15**	**-15.000**	**-15.000**	**-15.000**	0.0
g02	**0.803619**	0.803515	0.781975	0.726288	0.020
g03	**1**	**1.000**	**1.000**	**1.000**	0.00019
g04	**-30665.539**	**-30665.539**	**-30665.539**	**-30665.539**	0.00002
g05	**5126.4981**	**5126.497**	5128.881	5142.472	3.5
g06	**-6961.8138**	**-6961.814**	-6875.940	-6350.262	160
g07	**24.3062091**	24.307	24.374	24.642	0.066
g08	**0.095825**	**0.095825**	**0.095825**	**0.095825**	0.000000
g09	**680.6300573**	**680.630**	680.656	680.763	0.034
g10	**7049.25**	7054.316	7559.192	8835.655	530
g11	**0.75**	**0.750**	**0.750**	**0.750**	0.00008
g12	**1**	**1.000000**	**1.000000**	**1.000000**	0.0
g13	**0.0539498**	0.053957	0.057006	0.216915	0.031

differential evolution-based population [9]. The approach is applied to solve constrained optimization problems. Adding a belief space to the differential evolution algorithm, we were able to get a low computational cost while obtaining competitive results on a well-known benchmark adopted for evolutionary optimization. The main weakness of this approach is its apparent loss of diversity, due to its high selection pressure. Some of the knowledge sources of the belief space are designed to provide diversity, but more work remains to be done in this sense (as can be seen from the results obtained for g02). Some other directions of future work are the analysis of the impact of each knowledge source during the evolutionary process; or a comparison of different types of acceptance functions, which may allow a better exploration of the fitness landscape.

Acknowledgements

The first author acknowledges support from CONACyT to pursue graduate studies at the Computer Science Section at CINVESTAV-IPN. The second author gratefully acknowledges support from CONACyT through project 42435-Y.

References

1. Bentley, J.L., Friedman, J.H.: Data Structures for Range Searching. ACM Computing Surveys **11** (1979) 397–409
2. Chung, C.J., Reynolds, R.G.: A Testbed for Solving Optimization Problems using Cultural Algorithms. In Fogel, L.J., Angeline, P.J., Bäck, T., eds.: Evolutionary Programming V: Proceedings of the Fifth Annual Conference on Evolutionary Programming, Cambridge, Massachusetts, MIT Press (1996)
3. Chung, C.J., Reynolds, R.G.: CAEP: An Evolution-based Tool for Real-Valued Function Optimization using Cultural Algorithms. Journal on Artificial Intelligence Tools **7** (1998) 239–292
4. Coello Coello, C.A., Landa Becerra, R.: Adding knowledge and efficient data structures to evolutionary programming: A cultural algorithm for constrained optimization. In Cantú-Paz, E. et al., ed.: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2002), San Francisco, California, Morgan Kaufmann Publishers (2002) 201–209

5. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Publishing Company, Reading, Massachusetts (1989)
6. Iacoban, R., Reynolds, R.G., Brewster, J.: Cultural Swarms: Modeling the Impact of Culture on Social Interaction and Problem Solving. In: 2003 IEEE Swarm Intelligence Symposium Proceedings, Indianapolis, Indiana, USA, IEEE Service Center (2003) 205-211
7. Jin, X., Reynolds, R.G.: Using Knowledge-Based Evolutionary Computation to Solve Nonlinear Constraint Optimization Problems: a Cultural Algorithm Approach. In: 1999 Congress on Evolutionary Computation, Washington, D.C., IEEE Service Center (1999) 1672-1678
8. Kennedy, J., Eberhart, R.C.: Swarm Intelligence. Morgan Kaufmann Publishers, San Francisco, California (2001)
9. Landa Becerra, R., Coello Coello, C.A.: Culturizing differential evolution for constrained optimization. In: ENC'2004, IEEE Service Center (2004) Accepted for publication.
10. Price, K.V.: An introduction to differential evolution. In Corne, D., Dorigo, M., Glover, F., eds.: New Ideas in Optimization. McGraw-Hill, London, UK (1999) 79–108
11. Reynolds, R.G.: An Introduction to Cultural Algorithms. In Sebald, A.V., Fogel, L.J., eds.: Proceedings of the Third Annual Conference on Evolutionary Programming. World Scientific, River Edge, New Jersey (1994) 131–139
12. Reynolds, R.G.: Cultural algorithms: Theory and applications. In Corne, D., Dorigo, M., Glover, F., eds.: New Ideas in Optimization. McGraw-Hill, London, UK (1999) 367–377
13. Reynolds, R.G., Michalewicz, Z., Cavaretta, M.: Using cultural algorithms for constraint handling in GENOCOP. In McDonnell, J.R., Reynolds, R.G., Fogel, D.B., eds.: Proceedings of the Fourth Annual Conference on Evolutionary Programming. MIT Press, Cambridge, Massachusetts (1995) 298–305
14. Runarsson, T.P., Yao, X.: Stochastic Ranking for Constrained Evolutionary Optimization. IEEE Transactions on Evolutionary Computation **4** (2000) 284–294
15. Saleem, S .M.: Knowledge-Based Solution to Dynamic Optimization Problems using Cultural Algorithms. PhD thesis, Wayne State University, Detroit, Michigan (2001)
16. Storn, R.: System Design by Constraint Adaptation and Differential Evolution. IEEE Transactions on Evolutionary Computation **3** (1999) 22–34

An Approach of Student Modelling in a Learning Companion System

Rafael A. Faraco[1], Marta C. Rosatelli[2], and Fernando A. O. Gauthier[3]

[1] Grupo de Sistemas Inteligentes, Universidade do Sul de Santa Catarina,
Av. José Acácio Moreira, 787, Tubarão-SC, 88704-900, Brazil
rafael@unisul.br
[2]Programa de Pós-Graduação em Informática, Universidade Católica de Santos,
R. Dr. Carvalho de Mendonça, 144, Santos-SP, 11070-906, Brazil
rosatelli@unisantos.br
[3] Departamento de Informática e Estatística, Universidade Federal Santa Catarina,
INE-CTC, Cx.P. 476, Campus Universitário, Florianópolis-SC, 88040-900, Brazil
gauthier@inf.ufsc.br

Abstract. Nowadays there is an increasing interest in the development of computational systems that provide alternative (to the traditional classroom) forms of education, such as Distance Learning (DL) and Intelligent Tutoring Systems (ITS). Adaptation in the process of interaction with the student is a key feature of ITS that is particularly critical in web-based DL, where the system should provide real-time support to a learner that most times does not rely on other kinds of synchronous feedback. This paper presents the LeCo-EAD approach of student modelling. LeCo-EAD is a Learning Companion System for web-based DL that includes three kinds of learning companions - collaborator, learner, and trouble maker - that are always available to interact with and support the remote students. The student modelling approach of LeCo-EAD is appropriate to the DL context as it allows updating the student model in order to provide feedback and support to the distant students in real-time.

1 Introduction

Nowadays, there is an increasing interest in the development of computational systems that provide alternative forms of education, such as Distance Learning (DL) and Intelligent Tutoring Systems (ITS). ITS are able to (1) modify its knowledge bases by interacting with the students; (2) perceive the students' characteristics, preferences, and learning pace; and (3) adapt its pedagogical strategies to both individual and groups of students, according to the different learning situations.

Adaptation in the process of interaction with the student is a key feature of ITS. In order to provide adapted interactions, such systems need to keep updated about the state of the student knowledge level. This is accomplished by modelling the student. Student modelling and adaptation is particularly critical in web-based DL, where the system should provide real-time support to a learner that most times does not rely on other kinds of synchronous feedback (i.e., where other kinds of support usually take place in an asynchronous mode of interaction).

A kind of system that is considered an evolution of ITS and might be of relevance to DL are Learning Companion Systems (LCS). LCS were initially proposed by [6] and their architecture includes a learning companion (LC) as an additional component. The role of a LC is to be a virtual peer for the human student, interacting with him or her in order to collaborate with the learning process similarly as a real student does. This characteristic of LCS is appropriate to DL as the remote student usually accomplishes the learning activities alone.

This paper presents LeCo-EAD student modelling approach. LeCo-EAD is a LCS for web-based DL that includes multiples LCs that are always available to support the remote students. LeCo-EAD student modelling approach is appropriate to DL as it allows updating in order to provide real-time feedback and support to the distant students.

The paper is organized as follows. Section 2 discusses related work on student modelling in LCS. Section 3 presents an overview of LeCo-EAD. Section 4 describes LeCo-EAD student modelling components, functionality, and updating mechanism. Section 5 discusses the Artificial Intelligence (AT) technique used for student modelling and reports briefly on a formative evaluation of the system. Finally, Section 6 presents the conclusions.

2 Related Work

Among LCS there is no general, widely accepted model, or standard for the definition of the structure and functioning of student models. Usually, these are specially designed and built for each particular system, considering the domain; the number, kinds, and functionality of the LCs; and the computational constraints.

For instance, LECOBA [20] is a LCS developed in the domain of Binary Boolean Algebra and is based on the idea of inspectable student models. The system was implemented with the aim to show that less capable LCs could be beneficial for the learning of real students. In this system, the students have the chance to teach the LC, according to the approach of learning by teaching. In order to accomplish this, LECOBA provides a teaching window where the system presents the LC student model to the real student. This window aims to be a reflection tool, promoting thinking about the student knowledge on the domain. He or she is supposed to identify what the LC does not know and teach the LC these topics or concepts.

Another example is a reciprocal tutoring system in the domain of Lisp, in which a LC agent assumes the role of tutor and tutee in solving the problems [7]. The agents in the system can be real students or LC. When the LC behaves like a tutee, the system uses an overlay student model, and the agent pretends to learn in a level that is close to the real student level. When the system plays the role of tutor, the LC has a student model that keeps track of the student's actions.

According to Chan and colleagues [8], the student model in LCS can be of tree types: i) a single student model that provides information for the whole system; ii) a single student model and each learning companion has its own interpretation of the model; iii) each learning companion has its own student model.

Devedzic and Harrer [11] consider LCS in agent-based architectures of AI in Education systems that include pedagogical agents, by focussing on the co-learner pattern. The software pattern identified by the authors consists of a generalization of systems that were developed so far and have three main components: tutor, student, and co-learner. In these systems, each co-learner has its own student model. The advantage is a greater independence in the behaviour of the co-learners.

The student model in LeCo-EAD is a single one. Each LC has its own interpretation of this model and reacts in a different way, according to its role and/or type. Compared to the first approach pointed out by Chan et al. [8], this (second) type of student model is more flexible as it allows that each LC has its own behaviour, based on the same information about the student. Regarding the software pattern pointed out by [11] the complexity of updating several student models can be considered a constraint in the case of web-based systems like LeCo-EAD, as these might have to handle several students simultaneously connected to it.

3 LeCo-EAD Overview

LeCo-EAD is a LCS [9] that includes three kinds of LCs, which are always available to interact with the students. Each LC presents a particular kind of behaviour, handling different teaching strategies that are represented by the following types of LCs: collaborator [5], learner [20, 18], and trouble maker [1]. The LC that will interact with the student is, at first, chosen by the system, based on the student profile, which is identified through a Likert (or attitude) scale [16].

LeCo-EAD is domain-independent, as the contents of the didactic materials and the feedback messages can be easily replaced by the teacher and/or instructional designer to be used in different domains. The system architecture includes the modules of the traditional ITS (namely the domain module, tutor model, student model, and graphical user interface) with the addition of the LCs. Figure 1 presents LeCo-EAD architecture and its components are detailed below.

3.1 Domain Module

The domain module includes the contents that are presented to the student on the web pages and is structured as conceptual maps (see Figure 2 for an example). The objective of a conceptual map is to contextualize the subject matter attributing meaning to the topics studied [2, 4]. In LeCo-EAD the concepts included in the map should be studied one by one. A set of exercises is associated to each concept and the students are supposed to work on them. Depending on the student score, other concepts of the map are liberated, according to a pre-requisites structure. A degree of importance (given by a percentage) is attributed to each concept in this structure. The exercises and respective answers also possess relative weights: the knowledge factors (KFs). The KF is based on the certainty factor model [12] and represent the degree of importance of each exercise to a particular concept. That is, the higher the value of an exercise KF, the more representative this exercise is for mastering the related concept.

3.2 Tutor Model

LeCo-EAD tutor model handles the pedagogical decisions, which are made in accordance to the individual needs of each student. Thus, based on the pre-requisites structure, the tutor model manages the contents sequencing, deciding which topic should be presented to the student and when, and liberating the concepts represented in the conceptual map as the students reaches a satisfactory score to progress from one level to the next one.

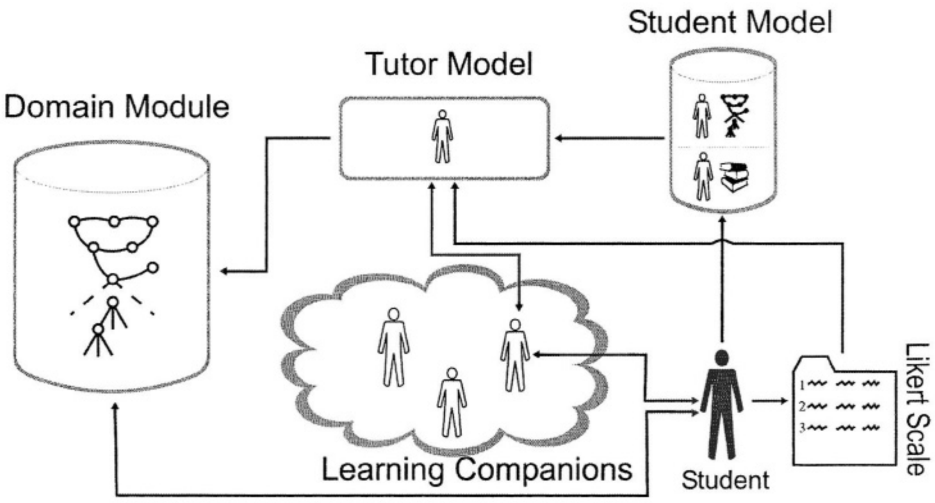


Fig. 1. The LeCo-EAD architecture

3.3 Student Model

Amongst the techniques used to represent the student knowledge, LeCo-EAD uses the perturbation or buggy student model [15]. This kind of model is an extension of the overlay model, which includes a subset of the expert's knowledge as well as possible student misconceptions. These misconceptions are represented by the exercises alternatives that are relative to each problem but are incorrect. The closer the weight of the alternative is to *-1.0,* the more it indicates a student lack of knowledge if he or she chooses this particular alternative as the exercise response. Thus, for each concept included in the conceptual maps, the system represents (1) if the student knows or does not know it and (2) a quantitative measure of how much he or she knows about it. Hence, besides mapping the correct knowledge, the system handles the student incorrect knowledge through the KF negative values in the various incorrect answers situations. Each student model includes information about the student performance in solving the exercises, the tracing of the student's accesses to the system, and the kind of LC attributed to the student based on the Likert scale.

3.4 Learning Companions

LCs are artificial students that may interact with real students in different ways: guiding, behaving like a co-learner or, yet, provoking the student. In LeCo-EAD each kind of LC (collaborator, learner, or trouble maker) has a graphical representation related with its behaviour and the student actively participates in the process of choosing his or her LC through a Likert scale [16]. The Likert scale indicates the students' predisposition in relation to objects, people, or events and was elaborated according to the following steps: (1) theory review, elaboration of the initial set of scale items, and reduction of items through a qualitative analysis; (2) collection of a pre-test sample, and pre-test; (3) statistics analysis; and (4) final scale elaboration [13]. The scale consists of an electronic form (questionnaire) that must be filled out by the student in the first time that he or she logs in the system. It aims to identify which kind of LC available in the system is appropriate to a particular student.

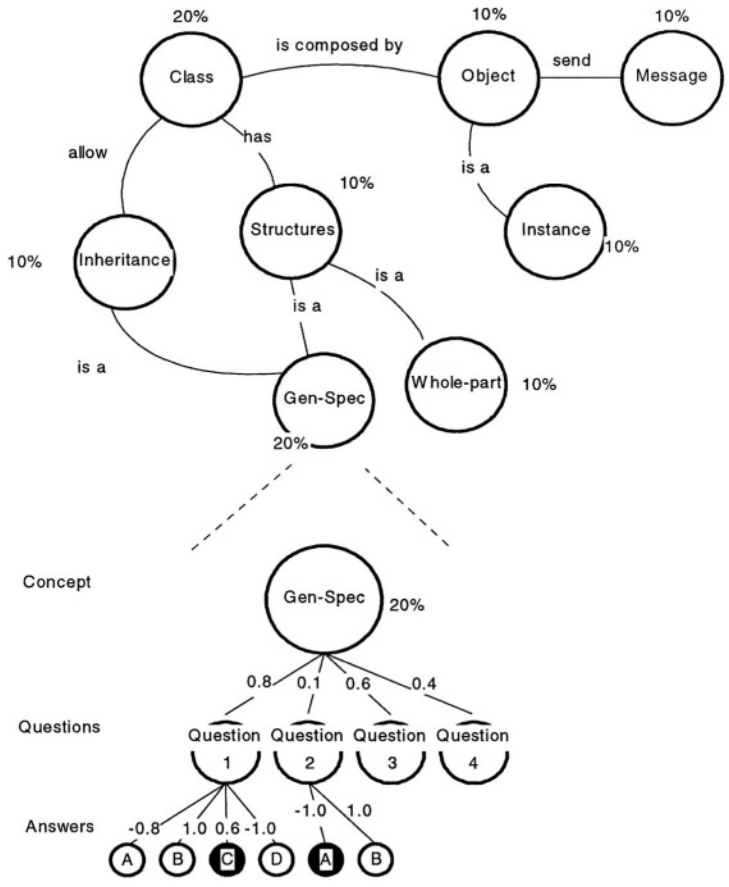


Fig. 2. A conceptual map in the domain of Object-Orientation

To each type of LC is associated a set of feedback messages that are presented to the student in two situations, namely: (1) when the system identifies that the student needs support to solve a problem, recognized when he or she exceeds a pre-established time limit to present an answer to an exercise; or (2) when the system identifies a misunderstanding in the answer given by the student.

4 Updating the Student Model

In LeCo-EAD, the student model is an overlay of the concepts, questions, and answers structure represented in the domain model. The process of updating the student model consists of updating the KFs based on the scores of the exercises presented in association with each concept and solved by the student. Updating the knowledge hierarchy is made by an algorithm that starts with the leaf-nodes (answer situation) and progress to the root (course) of the tree.

The manipulation of the KFs is based on rules of the kind **If** *evidence* **Then** *hypothesis*. There is a KF for the evidence (KF(evidence)) and for the hypothesis to be accepted (KF(hypothesis)). The degree of certainty [12] for rules of the kind evidence/hypothesis is given by equation (1):

$$KF(evidence\,/\,hypothesis) = KF(evidence) * KF(hypopthesis)\,. \qquad (1)$$

The rules of the kind evidence/hypothesis with respective KFs are applied to each pair answer/question situation of the contents hierarchy. The rules of the hierarchy presented on Figure 2 are detailed in Table 1.

Table 1. Rules of the hierarchy presented on Figure 2

Quest.1	**Rule1.1:** If the student chooses option "A" => (KF(A) = 1) Then the student almost does not know question => (KF(almost does not know) = - 0.8) KF(Rule1.1) = KF(A)*KF(almost does not know) = -0.8
	Rule1.2: If the student chooses option "B" => (KF(B) = 1) Then the student definitely knows question => (KF(definitely knows)= 1) KF(Rule1.2) = KF(B)*KF(definitively knows) = 1
	Rule1.3: If the student chooses options "C" => (KF(C) = 1) Then the student probably knows question => (KF(probably knows) = 0.6) KF(Rule1.3) = KF(C)*KF(probably knows) = 0.6
	Rule1.4: If the student chooses option "D" => (KF(D) = 1) Then the student definitely does not know question => (KF(definitely does not know) = -1) KF(Rule1.4) = KF(D)*KF(definitively does not know) = -1
Quest.2	**Rule2.1:** If the student chooses option "A" => (KF(A) = 1) Then the student definitely does not know question => (KF(definitely does not know) = -1) KF(Rule2.1) = KF(A)*KF(definitively does not know) = - 1
	Rule 2.2: If the student chooses option "B" => (KF(B) = 1) Then the student definitely knows question => (KF(definitely knows) = 1) KF(Rule2.2) = KF(B)*KF(definitively knows) = 1

The KF values used in the concepts hierarchy follow a linguistic representation according to Table2.

Table 2. Values attributed to linguistic meanings

Linguistic Meaning	Value
Definitely does not contribute/know	-1
Almost does not contribute/know	-0.8
Probably does not contribute	-0.6
Maybe does not contribute	-0.4
No information available	-0.2 to 0.2
Maybe contributes	0.4
Probably contributes/knows	0.6
Almost certainly contributes/knows	0.8
Definitely contributes/knows	1

As usually there are several questions related to a concept, the value of each rule referring to a particular concept should be incorporated as the student answers its related questions [12]. For handling multiple evidences referring to the same hypothesis, we consider:

$$KF(KF_1, KF_2) = KF_1 + KF_2 x(1 - KF_1), \qquad \text{If } KF_1 \text{ and } KF_2 \text{ are} > 0$$

$$KF(KF_1, KF_2) = \frac{KF_1 + KF_2}{1 - \min\{|KF_1|, |KF_2|\}}, \qquad \text{If } KF_1 \text{ or } KF_2 \text{ are} < 0$$

$$KF(KF_1, KF_2) = KF_1 + KF_2 x(1 + KF_1), \qquad \text{If } KF_1 \text{ and } KF_2 \text{ are} < 0$$

where:

KF_1 = Rule 1 knowledge factor;

KF_2 = Rule 2 knowledge factor; and

$KF(KF_1, KF_2)$ = Knowledge factor resulting from Rule 1 combined with Rule 2.

Intuitively, when the KF values are positive, there is an increase in the weight that results from the combination of the two rules. When they have different signs there is a decrease in the resulting weight, tending to zero. When the KF values are negative there is a decrease in the resulting weight, tending to *-1.0.* In any case, the resulting value of the KFs' manipulation ranges from *-1.0* to *1.0.*

Still considering the content hierarchy presented on Figure 2, updating the student model regarding the student knowledge about a concept (e.g., concept 1), is as follow:

1. The student model is initialized with zeros, indicating the student performance so far. That is, concerning LeCo-EAD, the student still does not know any concepts as he or she has not interacted with the system yet.

2. When the student chooses option "C" of question 1, Rule 1.3 is fired:
   KF(question1) = KF(C)*KF(probably knows) = 0.6
   KF(question1/concept1) = 0.6*0.8 = 0.048
3. When the student chooses option "A" of question 2, Rule 2.1 is fired:
   KF(question2) = KF(A)*KF(definitely does not know) = -1
   KF(question2/concept1) = -1 * 0.1 = -0.1
4. As there are two rules applied to a single concept:
   KF(question1/concept1) = 0.48
   KF(question2/concept1) = -0.1

   $$KF\ (concept\ 1) = \frac{0.48 + (-0.1)}{1 - \min\{0.48, 0.1\}}$$

   KF(*concept1*) = 0.38/0.9 = 0.42
5. The student model registers that the student masters concept 1 (Gen-Spec) with a value of *0.42.*

5 Discussion

The use of KFs in LeCo-EAD student model is based on the certainty factor model initially proposed for handling uncertainty in [3]. In spite of the fact that this model was criticised for not having a solid mathematical and probabilistic basis, it is computationally simple to implement, intuitive, and easy to manipulate [14].

In order to verify the adequacy of using KFs in LeCo-EAD, we carried out simulations considering the content hierarchies of the domain module with different number of levels. The first hierarchy considered was organized in four levels. The first level corresponded to the course and the levels below were units, concepts, and questions respectively. In the second case, we used a hierarchy that included only concepts and related questions. The certainty factor model did not present a satisfactory behaviour when the content hierarchy included many levels. The bigger the number of levels, the deeper the rules chaining mechanism has to operate. As a result, the final value of the knowledge accumulated for the student was distorted in deeper levels. In order to avoid this problem in LeCo-EAD inference mechanism, we limited the content hierarchy to two levels (as in the second type of contents hierarchy that we tested): concepts and related questions.

We believe that more robust techniques, such as Bayesian networks, could be used for modelling and handling the student knowledge. On the other hand, the computational complexity of this particular technique [10] might not be appropriate for a web-based learning system that has to provide real-time feedback to the user. In addition, Bayesian techniques depend strongly on prior probabilities, which are often not available, to initialize the student model [11].

Besides the process of updating the KFs for representing the student performance, LeCo-EAD also keeps track of the students' following actions within the system: the concepts that he or she studied, the questions answered, the time of access, the time of permanence at each web page, the LC initially indicated by the Likert scale, and if the student requests changing the type of LC.

In summary, student modelling in LeCo-EAD generates two main outputs: the first one refers to the student learning represented by the KF values, which vary between the range of *-1.0* (the student did not master any of the concepts) and *1.0* (the student demonstrates that mastered all the concepts). A value next to zero indicates that the student answered the questions with alternatives that do not measure efficiently if he or she knows or does not know a particular concept. Leaving an exercise response blank also has this same effect. The second kind of output regards keeping track of the student's actions, which prevents that LeCo-EAD repeats the concepts, exercises, and messages that were already presented to the student. Besides, all the interaction sessions of the student with the system are logged.

Regarding system evaluation, we carried out an empirical study as a formative evaluation, similarly as [17] and [19]. LeCo-EAD prototype was tested by a group of 5 teachers and 20 students of a Computer Science undergraduate course on Object Oriented Programming in Java. The aim was to test the system functionalities and the student model operation regarding providing the input for the system adaptation.

The formative evaluation brought to light some issues related with the system adaptivity. LeCo-EAD mechanism to choose the LC was considered sufficiently clear and direct by the empirical study participants, since the user participation in this process is explicit, either through the Likert scale or the feedback mechanism. This mechanism has a low demand upon the student in terms of the interaction: he or she just fills out the form of the attitude scale once in the beginning of the course, and afterwards controls the acceptance of the LC when closing the messages window. In addition, the conceptual map adaptation based on the student performance was considered appropriate and worked as expected.

6 Conclusions

In this paper we presented the student modelling approach of LeCo-EAD. LeCo-EAD is a LCS for DL. It includes three types of LCs that adopt different pedagogical strategies and are always available to interact with the students via the web.

The process of modelling the student in LeCo-EAD allows direct and quick updating, what is a desirable feature in web-based learning environments. It aims to monitor the student performance during the interaction with the didactic materials, keeping track of his or her actions. The former, i.e., monitoring the student performance, is carried out by updating the student model based on the inference mechanism of an expert system that uses the certainty factor model. The latter, i.e., keeping track of the student actions, is registered by the student model.

References

[1] Aimeur, E., Dufort, H., Leibu, D., Frasson, C: Some Justifications for the Learning by Disturbing Paradigm. In: Du Boulay, B., Mizoguchi, R. (eds.): Proc. of 8th International Conference on AI in Education. IOS Press, Amsterdam (1997) 119-126

[2] Ausubel, D.: Educational Psychology: A Cognitive View. Holt, Rinehart & Winston, New York (1968)

[3] Buchanan, B.G., Shortliffe, E.H.: Rule-Based Expert Systems. Addison-Wesley, Reading, (1984)
[4] Cañas, A.J., Ford, K.M., Brennan, J., Reichherzer, T., Hayes, P.: Knowledge Construction and Sharing in Quorum. In: J. Greer (ed.): Proc. of 7th World Conference on AI in Education. AACE, Charlottesville (1995) 228-225
[5] Chan, T.W.: Learning Companion Systems, Social Learning Systems, and Global Social Learning Club. International Journal of AI in Education 7(2) (1996) 125-159
[6] Chan, T.W., Baskin, A.B.: Learning Companion Systems. In: Frasson, C., Gauthier G. (eds.): Intelligent Tutoring Systems: At the Crossroads of AI and Education. Ablex Publishing Corporation, New Jersey (1990) Cap. 1
[7] Chan, T.W., Chou, C.Y.: Simulating a Learning Companion in Reciprocal Tutoring Systems. In: Proc. of Computer Support for Collaborative Learning'95 (1995). Available at: http://www-cscl95.indiana.edu/cscl95/chan.html.
[8] Chan, T.W., Chou, C.Y., Lin, C.J.: User Modeling in Simulating Learning Companions. In: Vivet, M., Lajoie, S. (eds.): Proc. of 9th International Conference on AI in Education. IOS Press, Amsterdam (1999) 277-284
[9] Chou, C.Y., Chan, T.W., Lin, C.W.: Redefining the Learning Companion: The Past, Present, and Future of Educational Agents. Computers & Education 40(3) (2003) 255-269
[10] Coper, G.F.: The Computational Complexity of Probabilistic Inference Using Bayesian Belief Networks. AI Journal 42 (1997) 393-405
[11] Devedzic, V., Harrer, A.: Architectural Patterns in Pedagogical Agents. In: Cerri, S. Gouarderes, G., Paraguaçu, F. (eds.): Intelligent Tutoring Systems. LNCS 2363. Springer-Verlag, Berlin (2002) 91-90
[12] Durkin, J.: Expert Systems: Design and Development. Prentice-Hall, New York (1994)
[13] Faraco, R.A., Rosatelli, M.C., Gauthier, F.A.: Adaptivity in a Learning Companion System. In: Proc. of the 4th IEEE International Conference on Advanced Learning Technologies. IEEE Press (2004) in press
[14] van der Gaag, L.C.: A Pragmatical View on the Certainty Factor Model. The International Journal of Expert Systems: Research and Applications 7 (3) (1994) 289-300
[15] Holt, P., Dubs, S., Jones, M., Greer, J.: The State of Student Modelling. In: Greer, J., McCalla, G. (eds.): Student Modelling: The Key To Individualized Knowledge-Based Instruction. Springer-Verlag, Berlin (1994) 3-35
[16] Likert, R.: A Technique for the Measurement of Attitudes. Archives of Psychology (1932) Vol. 140
[17] Meizalo, V., Torvinen, C., Suhonen, J., Sutinen, E.: Formative Evaluation Scheme for a Web-Based Course Design. In: Proc. of 7th Annual Conference on Innovation and Technology in Computer Science Education. ACM Press, New York (2002) 130-134
[18] Nichols, D.: Issues in Design Learning by Teaching Systems. AAI-AI-ED Technical Report n. 107. Computing Department, Lancaster University, Lancaster (1994)
[19] Tedesco, P.: MArCo: Building an Artificial Conflict Mediator to Support Group Planning Interactions. International Journal of AI in Education 13 (2003) 117-155
[20] Uresti, J.A.R.: Should I Teach My Computer Peer? Some Issues in Teaching a Learning Companion. In: Frasson, C., Gauthier, G., VanLenh, K. (eds.): Intelligent Tutoring Systems. LNCS 1839. Springer-Verlag, Berlin (2000) 103-112
[21] VanLehn, K., Niu, Z., Siler, S. Gertner. A.S.: Student Modeling from Conventional Test Data: A Bayesian Approach without Priors. In: Goettl, B.P, Halff, H.M., Redfield, C.L., Shute, V.J. (eds.): Intelligent Tutoring Systems. LNCS 1452. Springer-Verlag, Berlin (1998) 434-443

A BDI Approach to Infer Student's Emotions

Patricia A. Jaques and Rosa M. Viccari

PPGC - Instituto de Informática - Universidade Federal do Rio Grande do Sul, Bloco IV, Av. Bento Gonçalves, 9500 - Porto Alegre – RS - Brazil
{pjaques, rosa}@inf.ufrgs.br

Patricia had a Brazilian CNPq Scholarship

Abstract. In this paper we describe the use of mental states, more specifically the BDI approach, to implement the process of affective diagnosis in an educational environment. We use the OCC model, which is based on the cognitive theory of emotions and is possible to be implemented computationally, in order to infer the learner's emotions from his actions in the system interface. The BDI approach is very adequate since the emotions have a dynamic nature. Besides, in our work we profit from the reasoning capacity of the BDI approach in order to infer the student's appraisal, which allow us to deduce student's emotions.

1 Introduction

Due to the traditional dichotomy in the western society between reason and emotion, which was inherited from Descartes' dualist vision of the mind and body, little attention has been paid to the role of the affectivity in cognition. But, recent works of psychologists and neurologists have pointed out the important role of the affect in some cognitive activities, like decision taking [4] [5].

So, researchers of Artificial Intelligence in Education have considered the emotions in intelligent systems modelling, appearing thus a new field of research in AI: "Affective Computing". Picard [9] defines Affective Computing as "computing that relates to, arises from or deliberately influences emotions". We observe that the affective computing field is divided into two major branches of research interest. The first one studies mechanisms to recognise human emotions or to express emotions by machine in human-computer interaction. The second branch investigates the simulation of emotion in machines (emotion synthesis) in order to discover more about human emotions and to construct more realistic robots.

Moreover, due to the recent studies about the important role of the motivation and the emotions in learning [5] [16], techniques of affective computing have also been studied in order to model the emotions of the student in educational (computational) systems. To know the student's emotions help the agent to determine the more appropriate moment to intercede and to decide which kind of scaffold should the agent apply. If a lifelike pedagogical agent shows a sad emotive attitude when the student fails in the resolution of a task, it can make the student yet more anxious and depressed.

This paper presents a mental states approach, more specifically the BDI model, to recognise and model student's emotions in an educational environment. We infer satisfaction/disappointment, joy/distress, gratitude/anger and shame emotions according to OCC psychological model [8] from student's observable behaviour. More specifically, we profit from the reasoning capacity of the BDI approach to infer student's emotions from his actions in the system interface. The agent reasons about the student's actions and events in the educational system and to which emotion these events lead according to the student's goals.

This affective information about the student is used by an animated pedagogical agent, called Mediating Agent, that is responsible for motivating the student and promoting positive emotions in him, which fosters learning. This agent is part of the multi-agent architecture of an educational collaborative system. In this paper we focus only on the affective diagnosis using a BDI model, but more details about this affective agent can be found in [6].

2 Recognising and Modelling User's Emotions

In order for an affective computational system to interact effectively with the user, it must recognise the user's emotion to respond to him appropriately. In our work we catch the student's affective state from his observable behaviour, i. e. the student's actions in the system's interface. So, the agent obtains information about the student's emotions by analysing his actions. There are some examples of observable behaviour: success or failure in the execution of an exercise, ask for help and deny tutor's help.

In our work we recognise the *joy* and *distress, satisfaction* and *disappointment* emotions, as well as *gratitude, anger,* and *shame* emotions. The inference of these emotions is based on the OCC model [8]: a psychological model based on the cognitive approach of emotions that explains the origins of emotions by describing the cognitive processes that elicit them. Joy and distress emotions arise when the student is pleased because a desirable event happened (joy) or unpleased because an undesirable event did not happen (distress). For example, the student is displeased because he did not obtain a good grade in the course. The important point about *joy* and *distress* emotions is that they result from focusing only on the desirability or undesirability of the event. A person can also focus on other aspects of the events as well, for example that it was anticipated, or that some person was responsible for bringing it about. When this happens, different forms of emotions arise. This is the case of *satisfaction* and *disappointment* emotions. When the students focus on expected and suspected events and in the confirmation, or not, that these events will happen or happened; satisfaction and disappointment emotions can arise. For example, if the student expects to have provided a correct response to an exercise and it did not happen, he experiences disappointment emotion. The student can also focus on the agent that causes the undesirable/desirable event for him and, in this case, he will experience *gratitude* and *anger* emotion. The recognition of these two emotions helps the agent to estimate how helpful it is being for the student. Yet, the student can feel *shame* when the agent that caused the undesirable event is himself.

According to the OCC model, *joy* and *distress* emotions arise when a person focuses on the desirability of an event in relation to his goals. For example, for a determined student which has the goal of pleasing the teacher and his parents, obtaining a good grade is a desirable event. The OCC model defines joy as a person pleased about a desirable event, and distress as a person displeased about an undesirable event.

So, what we want to do, for the case of recognising joy and distress, is verify when an event of the educational environment is desirable for the student (according to his goals) and when the student is pleased because this desirable event happened or displeased because it didn't happen. This way, we need to define the *events* that can happen in the educational system, the *user's goal* (in order to know if the event is desirable or not) and how are we going to classify *an event as pleasant or not* in order to know if it elicits disappointed or satisfaction emotion?

Firstly, we defined some events that can arise in the educational system. Some examples of events are: the student didn't accomplish the task; the student provided a incorrect response for the exercise; and the student asked for help (due to space limitation, we just cite some examples of events).

Secondly, we need to determine the student's goals in order to verify the desirability of the events. But, *what goals does the student have in an educational situation*? According to Ames [1], students can have *mastery* or *performance* goals, which are the reasons for students engaging in learning and choosing to engage in academic tasks. Students who have a *learning/mastery goal* are oriented towards developing new skills and abilities, trying to understand their work, improving their level of competence, and learning new things. When students have *performance goals* they believe that performance is important and they want to demonstrate that they have abilities [1]. They feel successful when they please the teacher or do better than other students, rather than when they understand something new. In order to identify the student's goals we use the *Motivated Strategies for Learning Questionnaire* (*MSLQ*) [10]. The MSLQ is a self-report instrument that allows to determine students' goals and the learning strategies they use. It is based on a cognitive view of motivation and learning.

Thirdly, once we know the student's goals and the events that can arise in our educational system, we can determine the desirability of the events and also when the student is pleased/displeased with an event. This process is necessary to infer the student's appraisal, i. e., the cognitive evaluation that elicits emotions. This way, we classified the events according to their desirability based on what we know about student who have mastery or learning goals. With this information, we can determine student's emotion in our system. When the student is pleased about an expected desirable event that happened, he feels satisfaction emotion. When he is displeased because an expected desirable event didn't happen, he feels disappointment emotion.

2.1 Intensity of the Emotions

An emotion has always a determined intensity. According to the OCC model, the intensity of the emotions depends on some variables. The intensity of the *joy/distress* emotions depends mainly on the degree to which the event is *desirable* or not. The intensity of emotions *satisfaction/disappointment* also depends on the *desirability* of the event, on the *effort* made for the accomplishment of the event, and on the *realization* of the event (the degree to which the confirmed or disconfirmed event is realized).

The intensity of *gratitude* and *anger* emotions depends also on the *desirability* of the event.

Besides, the OCC theory considers that other global factors (that affect all OCC model's emotions) must also be considered: the event's *unexpectedness* (unexpected positive things are evaluated more positively than expected ones).

The degree of *desirability* of an event can be measured through the information that we have about performance and mastery oriented students. For example, we know that mastery oriented students desire more strongly to obtain a high grade. We use Soldato's model of effort [13] in order to measure the student's *effort* that can be minimal, little, medium, big and maximal. The *realization* variable can also be considered. For example, when an extrinsic student wants to obtain an excellent grade to please the teacher, if he just obtains a good grade, he achieves his goal partially. For the performance oriented student who usually receives average grade, to receive the maximal grade is an event with high *unexpectedness* and, so elicits the satisfaction emotion with a higher intensity. Due to the complexity of determining emotions' intensity by some cues from the student's observable behaviour, the current version of the prototype identifies just two degree of intensity: medium and high. But, in order to determine the emotions elicited and their intensity with more accuracy, the system foresees the insertion of other physiological sensors, such as skin conductivity and heartbeat sensors.

3 The BDI Model

The mental states approach describes an agent as an intentional system, i.e., having certain mental attitudes that are attributed to human beings, like "believe", "need", "desire", etc. This way, it is necessary to define which mental states are more appropriate. Bratman [3] proposed the BDI (Belief, Desire, Intention) model which is based on belief, desire and intention mental states.

The *beliefs* represent the information about the state of the environment that is updated appropriately after each sensing action. The beliefs can be viewed as the informative component of the system state.

The *desires* are the motivational state of the system. They have information about the objectives to be accomplished, i. e. what priorities or payoffs are associated with the various current objectives. The fact that the agent has a desire does not mean that the agent will do it. The agent carries out a deliberative process in which it confronts its desires and beliefs and chooses a set of desires that can be satisfied.

The *intention* is a desire that was chosen to be executed by a plan, because it can be carried out according to the agent's beliefs (because it is not rational that the agent carries out something that it does not believe). Plans are pre-compiled procedures that depend on a set of conditions for being applicable. The desires can be contradictory to each other, but the intentions can not. The intentions represent the currently chosen course of action. The intentions are persistent. An agent will not give up on its intentions – they will persist, until the agent believes it has successfully achieved them, it believes it can not achieve them or because the purpose of the intention is no longer present.

A rational agent will perform actions that it intends to execute without any further reasoning, until it is forced to revise its own intentions due to changes in its beliefs or desires. This may happen because of new events or the failure or successful conclusion of existing ones.

In our tutorial system, the agent's strategies and behaviour are described as the agent's beliefs. The decision of what to do and when to do it are the desires and intentions of the agent. This way, a determined strategy (belief) of the agent is activated if a desire of the agent becomes an intention.

In the next section, we present the BDI model and development tool used in this work: X-BDI.

3.1 X-BDI: The Logical Model Utilised

X-BDI (eXecutable BDI) is a BDI agent's model proposed by Móra [7]. The model can be also used as a tool for specification of BDI agents like the current formal models, as an environment for implementation and execution of agents. This way, it is not only an agent specification, but it may also be executed in order to verify the agent behaviour.

In order to reduce the distance between BDI agent's models and their implementation, instead of defining a new BDI logic or choosing an existing one and extending it with an operational model, Móra defines the notions of belief, desires and intentions using a logic formalism that is both well-defined and computational: *extended logic programming with explicit negation* (ELP) with the *well-founded semantics extended for explicit negation* (WFSX). ELP with WFSX extends normal logical programs with a second negation named explicit[1] negation. According to Mora [7], this extension allows to explicitly represent negative information (for example, a belief that a property *P* does not hold) and increases the expressive power of the language.

In the next section, we present the X-BDI tool. The goal is to describe how to specify and implement an agent, from a user's point of view, using X-BDI. More details about the X-BDI and the formalisms used to define the X-BDI model can be found in [7].

3.2 The Syntax of the X-BDI Tool

X-BDI is a tool for the implementation of an agent's cognitive module. In the case of this work it is used for the implementation of the Mind module of a pedagogical agent (the intelligent module of its architecture). Other modules of an agent (such as

[1] This is different of negation as failure or negation by default of normal logic programs [7].

sensors, effectors and interface) should be implemented using other programming languages and the X-BDI communicates with these other modules through *sockets*[2].

The beliefs (including actions) and desires must be specified in a file called *bdi.a,* which is loaded when the X-BDI begins its execution. The designer does not need to specify the agent's intentions, since the agent chooses its intentions through its desires. In the beginning of this file, the designer identifies the agent using the predicate *identity*(Agent_Name).

An *action* must be represented by the predicate *act:*

```
act (ag, action) causes effect if condition
```

where the attribute *ag* (agent's identification is optional). An action is composed of pre-conditions and pos-conditions. Pos-conditions represent the effects and consequences of an action and are represented by *effect.* The *condition* is a condition necessary to define a state or execute an action. The pre and pos-conditions can be expressed through the mental states of beliefs and desires.

Beliefs are represented by the predicate *bel* as follows:

```
bel (ag, p, t).
```

It means that the agent *ag* believes in a property *p* at a time *t.* The attribute ag e *t* are optional. If the attribute ag is not provided, it assumes that it is related to the described agent. If the attribute *t* is omitted, it assumes the current time.

Desires are described by the predicate *des:*

```
des (ag, p, t, prio).
```

It means that the agent *ag* desires the property *p* with the priority *prio* in the time *t.* Like in bel, the attributes *ag* and *t* area optional. The attribute *prio* is optional, but if specified it should have a value between zero and one.

Finally, the information received from the environment is described as follows:

```
[sense (property,time), sense (property,time), …]
```

4 Affective Recognition and Diagnosis Through Mental States

Let us see how the X-BDI cognitive kernel (the mind of our agent – the Mediating Agent) selects the affective tactics for the following scenario: the student has performance goals and he is disappointed because he provided an incorrect response to an exercise. The *cognitive kernel* receives the following information from the agent's sensors:

```
[current_time(2), sense(student_goal(performance), 1)].
[current_time(3), sense(event(not_correct_answer), 2),
                  sense(effort(high), 2)].
```

[2] A socket is one endpoint of a two-way communication link between two programs running on the network. A socket is bound to a port number in order to identify the application that data is destined to be sent [14].

The sensors notify the BDI cognitive kernel that the student has performance goals and his effort was high, and that an event happened - the student provided an incorrect response to the exercise.

So, the agent activates the desires "apply_tactics" and "emotion_sent" as intentions. The desire "emotion_sent" aims at sending to the Diagnostic Agent[3] the student's emotions. It uses this information for helping the Mediating Agent to choose the pedagogical tactics that are adequate in the cognitive and affective point of view. The desire "apply_tactics" is responsible for choosing the affective tactics that will be applied. The Mediating Agent was identified as "ag".

```
/* The agent's desires to apply an affective tactic */
des (ag, apply_tactics(Tactic), Tf, [0.6]) if
    bel(ag, choose_tactics(Tactic)).
act (ag, send_tactic(Tactic)) causes
    bel (ag, apply_tactics(Tactic)) if
        bel (ag, choose_tactics(Tactic)).
/* The Mediating Agent's desires to send the student's
emotions to the Diagnostic Agent */
des (ag, emotion_sent(Emotion,Intensity), Tf, [0.8]) if
     bel(ag, student_emotion(Emotion)),
     bel(ag, emotion_intensity(Emotion,Intensity)).
act (ag, send_emotion(Emotion,Intensity)) causes
     bel (ag, emotion_sent(Emotion, Intensity)) if
         bel (ag, student_emotion(Emotion)),
         bel (ag,emotion_intensity(Emotion,Intensity)).
```

In order for the agent to satisfy its intention of applying an affective tactic, it must accomplish the action of sending this tactic to the agent's actuator (*"send_tactic"* predicate) – the module of the Mediating Agent responsible for applying an affective tactic. To satisfy the intention *"emotion_sen"* it needs to send the emotion to the Diagnostic Agent (*"send_emotion"* predicate).

In order to send the emotions to the Diagnostic Agent, the Mediating Agent must know the student's emotions. It infers the student's emotions from the following beliefs:

```
/* The student is displeased with the event */
bel (ag, event_pleasantness(not_correct_answer,
                            displeased))  if
    bel (ag, student_goal(performance)),
    bel (ag, event(not_correct_answer)).
/* It is a prospect of an event */
bel (ag, is_prospect_event(not_correct_answer)) if
    bel (ag, event(not_correct_answer)).
```

[3] The Diagnostic Agent is another agent that composes the multi-agent architecture of the educational environment where the Mediating Agent is inserted. It has the goal of accomplishing the cognitive diagnostic and to choose the scafold tactics.

```
/* When the student is displeased, disappointment and
distress emotions arise */
bel (ag, student_emotion(disappointment)) if
     bel (ag,event_pleasantness(Event,displeased)),
     bel (ag,-is_mediador_action),
     bel (ag,is_prospect_event (Event)).
bel (ag, student_emotion(distress))  if
     bel (ag, event_pleasantness(Event,displeased)),
     bel (ag, -is_mediador_action).
```

The student is displeased with the event, because the event is undesirable, or it is desirable but it did not happen. When the student is displeased, it experiences distress emotion, and disappointment if it is the prospect of an event that was confirmed (*"is_prospect_event"* predicate). It is the case of the event "not_correct_task_answer", since when the student accomplishes a task he has an expectation that this event would happen. To elicit disappointment and distress emotions the event should not be caused by the Mediating Agent. The agent's actions elicit emotions as anger and gratitude.

It is also important to verify the value of the variables that affect the emotion's intensity:

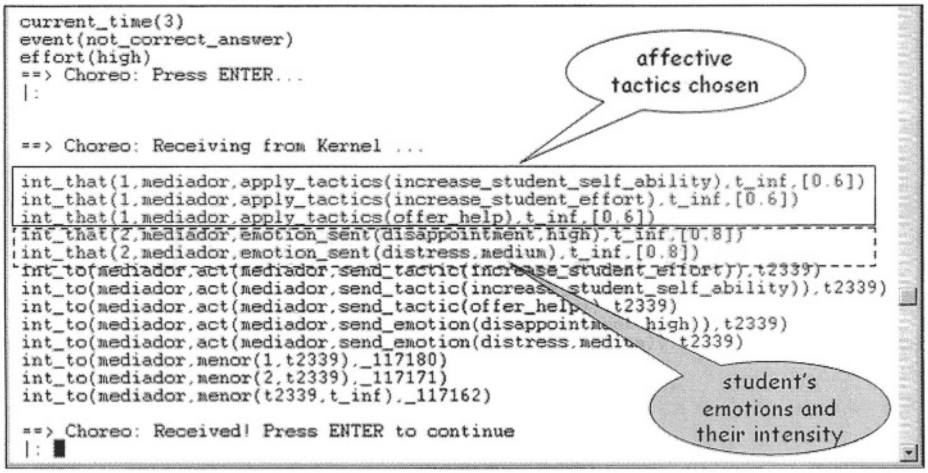
```
bel (ag, emotion_intensity(disappointment, high)) if
     bel (ag, effort(high)),
     bel (ag, student_emotion(disappointment)).
```

The variables that affect the emotion's intensity are effort, realization, unexpectedness and undesirability for disappointment, and undesirability for distress. If **one** of these variables has a higher value (marked with high) the student experiences the specific emotion with high intensity, otherwise he experiences emotions with medium intensity. The values of the variables that affect the emotion's intensity are sent by the sensor of the body module and these values can be medium or high. It is responsible for identifying the value of these variables with questionnaires and student's observable behaviour.

Finally, the agent chooses the tactics through the beliefs showed below. The affective tactics are: (1) to increase the student's self-ability, (2) to increase the student's effort; and (3) to offer help to the student. Once it chose the affective tactic, it can accomplish the action of sending the tactic to the actuator module. As this action is the restriction for the elected intention to be satisfied, the agent's intention of applying an affective tactic is accomplished.

```
bel (ag, choose_tactics(increase_student_self_ability))
if   bel (ag, student_emotion(disappointment)),
     bel (ag, event(not_correct_answer)),
     bel (ag, student_goal(performance)).
bel (ag, choose_tactics(increase_student_effort)) if
     bel (agent, student_emotion(disappointment)),
     bel (ag, event(not_correct_answer)),
     bel (agent, student_goal(performance)).
bel (ag, choose_tactics(offer_help))   if
     bel (agent, student_emotion(disappointment)),
     bel (ag, event(not_correct_answer)),
     bel (ag, student_goal(performance)).
```

The inference of the student's emotions and the choice of the affective tactics by the X-BDI kernel can be visualized in the interface of the Prolog that is shown in *Fig. 1.* In this case, in order to show how the X-BDI kernel works, we programmed it to show the result in the Prolog Interface, instead of sending this information to the body of the agent (the way that the agent should behave). The set of intentions chosen by the agent's mind for the example previously described are represented by the predicates *int_that* inside the shaded squares in *Fig. 1.* In the dark gray square we can see the emotions that were chosen by the intention *emotion_sent.* The light gray square shows the tactics that were chosen by the intention *apply_tactics.* The predicates *int_to* indicate the actions that are made by the agent "ag" (Mediating) in order to satisfy the intentions.



```
current_time(3)
event(not_correct_answer)
effort(high)
==> Choreo: Press ENTER...
|:


==> Choreo: Receiving from Kernel ...
int_that(1,mediador,apply_tactics(increase_student_self_ability),t_inf,[0.6])
int_that(1,mediador,apply_tactics(increase_student_effort),t_inf,[0.6])
int_that(1,mediador,apply_tactics(offer_help),t_inf,[0.6])
int_that(2,mediador,emotion_sent(disappointment,high),t_inf,[0.8])
int_that(2,mediador,emotion_sent(distress,medium),t_inf,[0.8])
int_to(mediador,act(mediador,send_tactic(increase_student_effort)),t2339)
int_to(mediador,act(mediador,send_tactic(increas[illegible]student_self_ability)),t2339)
int_to(mediador,act(mediador,send_tactic(offer_hel[illegible] t2339)
int_to(mediador,act(mediador,send_emotion(disappointm[illegible]high)),t2339)
int_to(mediador,act(mediador,send_emotion(distress,medi[illegible]t2339)
int_to(mediador,menor(1,t2339),_117180)
int_to(mediador,menor(2,t2339),_117171)
int_to(mediador,menor(t2339,t_inf),_117162)

==> Choreo: Received! Press ENTER to continue
|: █
```

Fig. 1. The choice of the affective tactics by the X-BDI cognitive kernel

5 Conclusions

In this paper we described the use of the BDI approach for the implementation of the mind of an affective pedagogical agent that infers student's emotions, models these emotions and chooses an appropriate affective tactic according to these emotions. The choice of the mental states approach for this implementation is based on the cognitive approach of emotion [12] which considers that emotions are elicited by a cognitive evaluation of the personal significance of an agent, object or action (appraisal). For example, a person feels fear of being bitten by a snake because he evaluates that this event (the bite) can have an undesirable consequence for him (he can die or be seriously hurt). This way, the agent deduces the affective state of the student through a BDI reasoning which aims at discovering the cognitive evaluation (his appraisal) made by the student. To do so, it needs to know the events that are happening and the student's goals.

Besides, the affective model must be dynamic enough to consider the changes in the emotional states [2]. Since the motivation and the affectivity of the student may vary in a very dynamic way (the student may not feel satisfied at some determined moment and feel more satisfied in another one), the use of the BDI approach for the implementation of the student model show to be very convenient, because it allows simple revisions and frequent modifications of the information about the student [2]. The student model is built dynamically from each interaction in real-time.

The BDI approach has been used by our research group at UFRGS as tool for modelling the cognitive abilities of the student (Giraffa's and Viccari work [15]) and also his motivation and the affective state displeased (Bercht's work [2]). Our work differs from Bercht's work [2] in the methodology used to recognise the student's emotions. In Bercht's work, the inference of the student's appraisal was made by an expert that inserted the related rules as beliefs of the agent. In this case, events are mapped directly to emotions. In our work the inference of student's appraisal is made by the agent itself. An advantage of our proposal is that it is not necessary for an expert to determine all the rules for the student's affective state inference in order to implement them in the agent in advance. The agent deduces the student's emotions by reasoning about his appraisal from the information that it has about the student.

We see that a future work can be to extend the belief-desire-model, more specifically the X-BDI (the BDI tool used in the implementation of this thesis), in order to also include personality traits, emotions, and moods. According to de Rosis [11], this approach offers several advantages. The first one is that it opens the opportunity of driving consistent behaviours of agents from a model of their cognitive state: the system of beliefs, desires, and intentions may trigger emotions, regulate the decision of whether to show or to hide them, and finally, drive externalized actions. In this case, we are incorporating an architecture of emotions (emotion synthesis) in the agent in order for it to generate affective behaviour more consistent and believable.

References

1. Ames, C. Motivation: What Teachers Need to Know. Teachers College. 91, 3, 409-21, 1990.
2. Bercht, M; Viccari, R. Pedagogical agents with affective and cognitive dimensions. In: Congreso Iberoamericnao de Informatica Educativa, 2000, Vina del Mar.
3. Bratman, M. E. What is intention? In: Cohen, P., Morgan, J., and Pollack, M. (Eds.) Intentions in Communication. pp. 15-31. MIT Press, Cambridge, MA, 1990.
4. Damasio, A. Descartes' Error. New York: G. P. Putnam, 1994.
5. Goleman, D. Emotional Intelligence. New York: Bantam Books, 1995.
6. Jaques, et al. Applying Affective Tactics for a Better Learning. European Conference on Artificial Intelligence, 2004. To be presented.

7. Móra, M. C. et al. BDI models and systems: reducing the gap . ATAL'98. Paris: 1998.
8. Ortony, A. et al. The cognitive structure of emotions. Cambridge Univ. Press, UK, 1988.
9. Picard, R. Affective Computing. Cambridge: MIT Press, 1997. 262 ps.
10. Pintrich, P. et al. A Manual for the Use of the Motivated Strategies for Learning Questionnaire. TR 91-B-004. The Regents of the University of Michigan. 1991.
11. de Rosis, F. Toward Merging Cognition and Affect in HCI. Applied Artificial Intelligence, Philadelphia, v. 16, n. 7-8, p. 487-494, August 2002.
12. Scherer, K. R. Appraisal Theory. In: T. Dalgleish; M. Power (Eds.). HandBook of Cognition and Emotion. John Wiley & Sons Ltda, 1999.
13. del Soldato, T.; de Boulay, B. Implementation of Motivational Tactics in Tutoring Systems. In: Jounal of Artificial Intelligence in Education, v. 6, nro. 4, 1995. p. 337-378.
14. Sun. Custom Networking: all about sockets, http://java.sun.com/docs/books/tutorial/networking/sockets/definition.html. November, 2003.
15. Viccari, R. M. et al. Towards a New Perspective for ITS: Using a Mentalistic Approach to Model Cognitive Pedagogical Agents. International Journal of Continuing Engineering Education and Life Long Learning. Osaka, Japão, v.9, 1999.
16. Vygotsky, L. S. The Problem of the Environment. In: The Vygotsky Reader. Cambridge, MA: Blackwell, 1994. pp. 338-354.

Mobile Robotic Supported Collaborative Learning (MRSCL)*

Ruben Mitnik, Miguél Nussbaum, and Alvaro Soto

Departamento de Ciencia de la Computación, Escuela de Ingeniería,
Pontificia Universidad Catolica de Chile,
Vicuña Mackenna 4860, Santiago, Chile
rmitnik@puc.cl, {mn, asoto}@ing.puc.cl

Abstract. In this paper we describe MRSCL Geometry, a collaborative educational activity that explores the use of robotic technology and wirelessly connected Pocket PCs as tools for teaching and reinforcing concepts of basic geometry. The application can be considered robotic aided education since the robot acts as a mediator in the learning experience while the students are able to learn concepts that are not related to Robotics. One mayor difference with previous computer based teaching tools is that the robot motions are not absolute, but relative to external references and past moves mapped in the real world. Furthermore, MRSCL Geometry helps students to develop social abilities, such as communicating with their pairs, and geometrical abilities, such as interpreting and describing information about lengths, positions, and trajectories. Autonomous robot navigation is carried out by developing the teaching activities in a simplified world consistent of clear space and bright color landmarks that are used by the robot as the main reference to continuously track its position. In this respect, we develop vision algorithms for the real time detection and tracking of landmarks. We also implement distance estimation and control algorithms to allow the robot to move and turn accurately.

1 Introduction

Historically, there have been evident problems in the teaching and learning of school geometry. One of these problems is the algebraisation of geometry, along with the suppression of geometrical thinking. This issue uncovers, for example, when angle related problems are transformed into algebraic exercises of "solving for the unknown". A direct consequence of the algebraisation of school geometry is that calculation and algebraic manipulation become the main focus of the learning activity, leaving aside the development of geometrical abilities, such as visual reasoning [5]. In relation to measuring concepts and according to [8], it is necessary to understand the logic of measurement before being able to use a standard measuring instrument, such as a ruler. In this respect, a crucial concept

* This work was partially funded by FONDECYT grant 1040605.

in the process of measurement learning is the development of the "measurement sense", defined as the ability to estimate lengths and draw lines of a given size [3]. The construction of this "mental ruler" constitutes a fundamental issue in the development of the "measurement sense", becoming an internal measuring instrument of the student. According to [3], measurement is one of the most important real world mathematical applications, since it bridges two critical realms of Mathematics: geometry as spatial relations and real numbers.

New technologies allow the development of educational software devoted to geometry and measurement teaching. According to Hoyles and Noss [5] [6], a key issue for this type of software is the fact that it must provide observers a "window" onto the mathematical meaning under construction. This means that while the students use the software to construct models by which they explore and solve problems, their thoughts become simultaneously externalized and progressively shaped by their continuous interaction with the program. Logo environment is the most common tool used to build such software. A positive characteristic of Logo environment is the fact that it provides instant visual feedback regarding the received instructions. Nevertheless, these virtual settings lack mobility and 3D observation, deficiencies which are not present in real world environments.

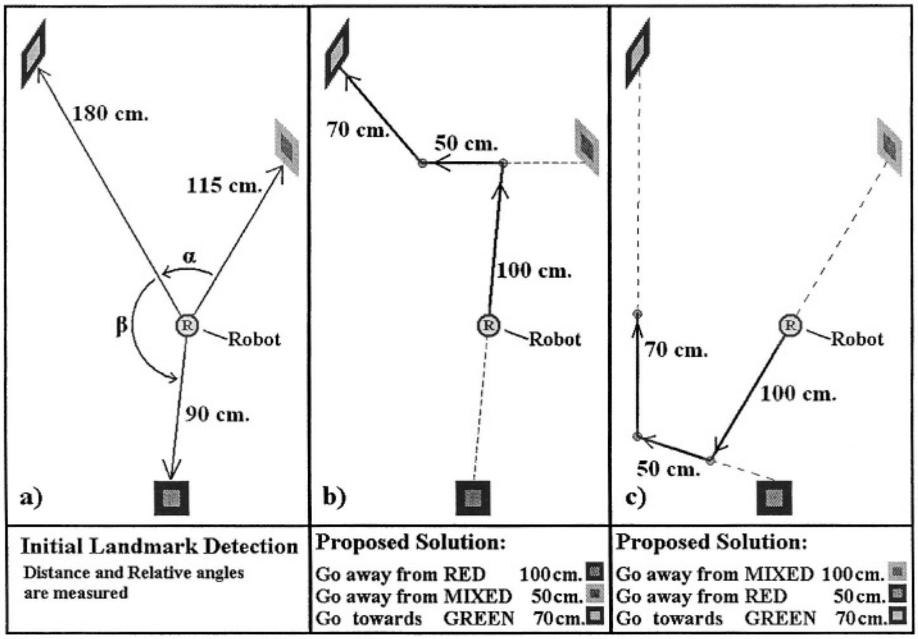
In the area of Robotics, while there have been applications devoted to education, these have been mainly focused on teaching subjects closely related to the Robotics field, such as programming [10] [11], robot construction [1] [11], and artificial intelligence [1], among others.

Our previous research has been focused on the usability and learning benefits of technologies inside the classroom. Particular emphasis has been made on Mobile Computer Supported Collaborative Learning (MCSCL). We have shown that MCSCL solves problems of coordination, communication, organization, negotiation, interactivity, and lack of mobility detected in collaborative activities for 6-7 year old students [13]. MCSCL has also proved to provide a highly motivating learning environment in high schools, changing the classroom dynamics and promoting collaboration among students [4]. Moreover, MCSCL introduces a space that favors constructivism in order to achieve the creation of new knowledge in a reflexive process directed by the teacher [12].

Mobile Robotic Supported Collaborative Learning (MRSCL) introduced in this paper arises from the addition of robotic technologies to MCSCL. The particular MRSCL activity presented in this work, MRSCL Geometry, responds to the need for better tools that can help to teach and reinforce fundamental geometrical concepts such as advance and retreat and length estimation. Moreover, MRSCL Geometry helps students to develop social abilities, such as communicating with their pairs, and geometrical abilities, such as interpreting and describing information about lengths, positions and trajectories. MRSCL Geometry also innovates with respect to previous applications of Robotics technology in education in the following features: the robot is not the aim but the mean; students have a high degree of mobility; the learning is constructivist [2] and develops on a collaborative environment [7]; the learning is incidental by allowing the students to be active actors in the activity [9]; the robot is completely autonomous using sensing information to guide its way. This paper mainly focuses on the architectural and algorithmic implementation.

2 MRSCL Geometry

The application consists of three students who have to solve with a robot, in a collaborative way, a geometry problem. The components of MRSCL Geometry are: an autonomous robot, three different landmarks (shown in the screenshots of Figure 2), three Pocket PCs (handhelds), and a set of different sized rulers. The handhelds and the robot are wirelessly connected. The activity is executed over the floor or any other flat horizontal surface with enough space to contain it.



Initial Landmark Detection
Distance and Relative angles are measured

Proposed Solution:
Go away from RED 100cm. ▣
Go away from MIXED 50cm. ▨
Go towards GREEN 70cm. □

Proposed Solution:
Go away from MIXED 100cm. ▨
Go away from RED 50cm. ▣
Go towards GREEN 70cm. □

Fig. 1. a) The initial landmark setting is detected by the robot, b) Path followed by the robot when the correct answer (in the lower part of the figure) has been proposed. c) Path followed for another (erroneous) answer

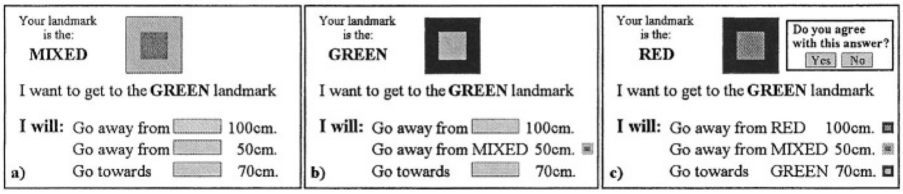
The main idea behind MRSCL Geometry is that the students help the autonomous robot to arrive to a predefined place determined by it. In doing so, they must collaborate with each other, and at the same time learn and practice important geometrical concepts such as relative positions and distance measurement. To achieve this goal the robot can only move in linear paths, either towards or away from certain landmarks. The role of the students is to indicate the robot in relation to which landmark it must execute each movement in order to arrive to the predefined goal. For this, the students can use whatever rulers they need from the available set.

Figure 1 shows an example of MRSCL Geometry. First, the robot detects its distance to the landmarks and the relative angle between each pair of them (Figure 1a). Next, it generates the problem autonomously and sends it to the students. Through it, the robot tells them he is first going to move 100 cm. away from some landmark, next he is moving 50 cm. away from a second landmark, to finally move 70 cm. towards the last landmark. He also tells the students that the goal is for it to arrive to the green landmark. The students have to find out which is the right sequence of landmarks, each student being in charge of one landmark, specified on the screen of his mobile device. After the students determine the landmark sequence to follow, the robot executes the solution proposed by them. Figures 1b and 1c show the paths followed by the robot for two possible answers (shown in the lower part of each figure). It can be seen that the answer in Figure 1b corresponds to the correct solution of the problem since the robot arrives to the green landmark. Figure 1c in the other hand corresponds to an incorrect solution.

2.1 MRSCL Geometry Main Activity

Each student is given a landmark and a handheld. First, the students must freely position their landmarks around the robot at about one to two meters from it. After this initial positioning, landmarks are not moved except by explicit request from the robot. Next, the robot rotates around its center searching for the landmarks. Once detected, the landmarks act as an external frame of reference for the robot. Using these references the robot can establish at any time its exact position within the frame just by measuring its distance to the landmarks. The landmarks also act as references for the robot motions, particularly for those where it must move away or towards them, as required by the activity. Next, according to the position of the landmarks, the robot generates a problem. Each problem consists of a goal corresponding to a landmark, to which it wants to arrive, and a sequence of three motions, each consisting of a direction (away or towards) and a distance. Since the students are learning measurement concepts, and to facilitate the creation and development of their mental ruler [3], we use multiples of ten as the magnitude for the motion distances and a maximum distance per movement of 2 meters.

Once the robot has generated the problem, it assigns each handheld the reference of a different landmark. The assigned landmark and the problem are then transmitted wirelessly to each handheld which displays this information visually on the screen (Figure 2a). The simplicity of the computer interface responds to the importance of centering the attention of the students on their common real-world coordinate plane rather than on the handheld itself. In this way, interaction and collaboration arise in a natural way. It also allows students to express and show their ideas spatially, enabling a better understanding among them. Each student is allowed to select one of the three movements, indicating the robot to execute it in relation to his assigned landmark. When a student selects a movement, his choice is shown visually in every handheld, disabling the movement for subsequent selections. Figure 2b shows a screenshot of the handheld assigned to the green landmark. The second movement has already been selected in the handheld assigned to the mixed landmark. As it can be seen, at this point the handheld assigned to the green landmark can no longer select the second movement.


a)
Your landmark is the: MIXED
I want to get to the **GREEN** landmark
I will: Go away from 100cm.
Go away from 50cm.
Go towards 70cm.

b)
Your landmark is the: GREEN
I want to get to the **GREEN** landmark
I will: Go away from 100cm.
Go away from MIXED 50cm.
Go towards 70cm.

c)
Your landmark is the: RED
Do you agree with this answer? Yes No
I want to get to the **GREEN** landmark
I will: Go away from RED 100cm.
Go away from MIXED 50cm.
Go towards GREEN 70cm.

Fig. 2. a) Screenshot of the handheld assigned to the mixed landmark upon receiving the problem, b) Screenshot of the handheld assigned to the green landmark after the second movement has been selected in (a), c) Screenshot of the handheld asssigned to the red landmark after the three students have chosen their moves. The pictures shown in each screenshot correspond to the real landmarks used in MRSCL Geometry. The inner and outer color square design was chosen for the robot to achieve a robust visual detection

Once all the students have selected a movement, the robot asks each student individually if s/he agrees with the proposed solution (Figure 2c). If any student answers “no” the message “Come to an Agreement” displays on the handhelds, after which they reset to the state shown in Figure 2a. The consensus requirement forces the group to discuss their ideas and points of view. Cortez et al. [4] proves that this required consensus stimulates description and explanation of ideas in a face to face interaction. This idea, in addition to concept externalization by each student, provides the “window onto mathematical meanings” described in [5]. Throughout debate students externalize thoughts, allowing the observation of the whole process of creation, evolution, change, and maturation of concepts. After consensus has been reached, the robot executes the proposed solution. In this way the students get 3D feedback of their hypothesis, measurements, and estimations. If the solution is not correct, the robot returns to its initial position restarting the problem (Figure 2a). If the solution is correct, the robot asks the students if they want to continue playing. If this is the case, the robot moves randomly to a different starting position, asks the students to reposition the landmarks, and generates a new problem.

3 Robot Autonomous Navigation

Robust autonomous navigation has been a main research topic for the Robotics community during the last decades, being mapping, localization, and path planning, some of the main research issues. In MRSCL Geometry we solve this problem by developing the activities in a simplified world (playground) consisting of clear space and bright color landmarks that are used as the main references to continuously track the position of the robot (see Figures 1 and 2).

By sensing the distance from the robot to a specific landmark and detecting its bearing angle to the set of landmarks, the robot can attain at each moment an estimation of its absolute position in the playground. Using this estimation the robot can take adequate control actions and lead its way to complete the learning activities. In this Section we describe the main features of the robot sensing capabilities, while in Section 4 we describe the details of the motion control scheme.

3.1 Robot Range Sensing

We use an off-the-shelf sonar to measure distances. In addition, we use a Kalman filter to improve the accuracy of real time distance estimation during movements. The only state variable corresponds to the correct distance to the landmark measured.

3.2 Robot Visual Perception

The purpose of the vision algorithms is to detect and follow in real-time some predefined landmarks. The landmarks consist of a colorful square (main color) enclosed by a larger square of a secondary color (see Figures 1 and 2). Due to hardware limitations the camera on the robot transmits only binary images. Given a RGB maximum and minimum, the transmitted binary image corresponds to the pixels contained in such RGB range. We will define "main color" and "secondary color" as the colors of the inner and outer squares of the landmark respectively. The main image and "secondary image" are the binary images acquired using the main and secondary color ranges, respectively.

Detection Algorithm. This algorithm determines whether the camera is perceiving a particular landmark. The algorithm is shown in Figure 3a, where the right hand images correspond to the outcome of each step. First, the camera acquires the main and secondary images according to the corresponding colors of the requested landmark. Next, it segments the main image into regions, defined as groups of connected pixels. Noise filtering eliminates regions with smaller area than the minimum predefined acceptance area. Subsequently the Secondary Color Enclose Index (SCEI) is calculated for each region. This index corresponds to the fraction of the contour of the region whose color corresponds to the secondary color. The SCEI of each region is calculated by dividing the number of detected secondary pixels of its contour by the total amount of contour pixels. Finally, the region with greater SCEI is selected. If its SCEI is greater than the minimum predefined SCEI acceptance value the region is detected as the landmark. In this case, the algorithm returns the position of the horizontal center of mass of the region (vertical line in last image).

The implemented algorithm completes three detections per second under any visual condition. When visual distractions were placed, less than 1% of the detections proved false. Even though the algorithm proved robust, real landmarks were not detected nearly 1% of the times.

Tracking Algorithm. This algorithm allows real time tracking of a detected landmark without the need to execute a complete new detection, thus diminishing the processing time. Figure 3b shows the algorithm with the corresponding outcome of each step. After the landmark has been detected, a binary image is stored. This image corresponds to the region of the inner square of the landmark. A new image of the color of the inner square (main color) is then acquired, segmented, and filtered. Subsequently the matching index of each region is computed. This index is defined as the number of pixels in the region that match with the landmark image. In the example of Figure 3b the landmark image, represented by gray dots, has been placed over the main segmented image. This illustrates, for example, that region 4 has 8 matching pixels while region 3 has none. Finally the region with the greatest matching index is assumed to be the new position of the landmark, updating the landmark image for the next iteration.

A drawback of the algorithm, due to its recursive nature, is that if by any chance it tracks an erroneous object, it will continue to follow that object, completely loosing the landmark. Nevertheless, this problem was not observed during execution. Experimentally, this algorithm performs 8-10 tracking iterations per second, limited by the camera transmitting speed.

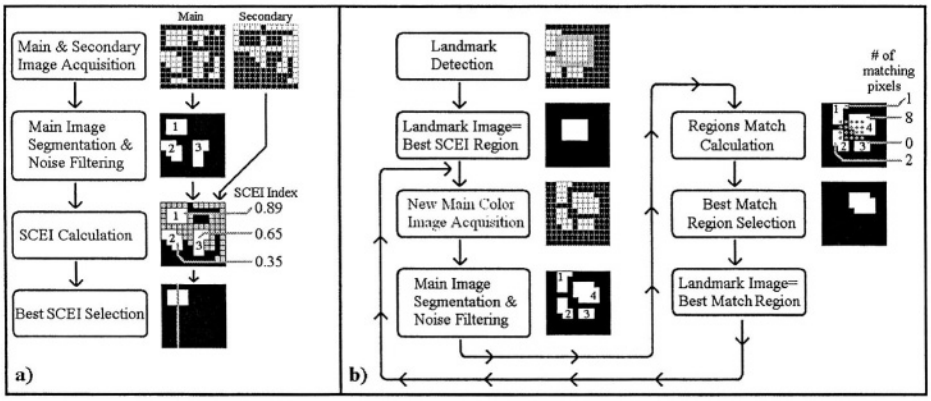

Main
Secondary
Main & Secondary Image Acquisition
Main Image Segmentation & Noise Filtering
SCEI Calculation
Best SCEI Selection
SCEI Index
0.89
0.65
0.35
a)
Landmark Detection
Landmark Image= Best SCEI Region
New Main Color Image Acquisition
Main Image Segmentation & Noise Filtering
Regions Match Calculation
Best Match Region Selection
Landmark Image= Best Match Region
# of matching pixels
1
8
0
2
b)

Fig. 3. a) Detection algorithm, b) Tracking algorithm

4 Implementation

4.1 Architecture

The architecture of MRSCL Geometry is shown in Figure 4a. WiFi communication cards are used on Pocket PCs to permit the activity to be self contained. The Robot also has a Pocket PC with WiFi, which acts as Robot Controller and communication media with the three students.

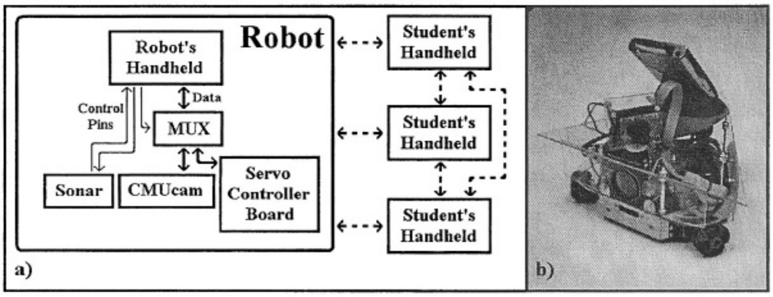


Fig. 4. a) Architecture of the robot. b) The robot

4.2 Robot

The robot we use is the omnidirectional Palm Pilot Robotic Kit (PPRK). Vision is accomplished using the CMUcam which has RGB resolution of 80 x 143 pixels. It uses serial communication with a maximum transmission speed of 115.200 bauds, not allowing real time color video. Nevertheless, the cam can transmit binaxy images of a given the RGB range at a rate of 17 frames per second. In this mode resolution decreases to 80 x 44 pixels. The camera is mounted on a servo to allow independent rotation.

4.3 Control Systems

For the correct performance of the application, two control systems where developed. The first controls the trajectory of the robot while the second controls the angular rotation to achieve an accurate relative angle measurement.

Linear Movement Control. The purpose of this control is to maintain the robot moving in a straight direction while advancing or retreating Because of the omnidirectionality, the robot can rotate while moving. First, the robot detects and centers the corresponding landmark in its field of view (FOV). Next it begins moving, either forward or backwards according to the activity. During its motion the robot continuously tracks the landmark keeping it always in the center of its FOV. To determine if it has reached its final position, the robot continuously measures its distance to the landmark. The efficiency of the control depends on the speed at which the vision module runs the landmark tracking algorithm.

Rotation Control. This algorithm controls the robot and the rotation of the camera in order to estimate accurately the relative angle between landmarks. Because the estimation of the angular speed is not accurate enough, the servo that supports the camera is used to measure the traveled angle. The algorithm is divided into two stages: "robot search" (stage 1) and "cam search" (stage 2). Initially, the robot rotates searching for the required landmark. When detected, it centers the landmark in its FOV. During this first stage, the position of the camera remains unchanged in relation to the robot. Next, the algorithm passes to the "cam search" stage where the robot stays halted while the camera rotates searching for the next landmark. Once detected, the camera centers on it, measuring the angle difference of its supporting servo motor. The accuracy of the measurement corresponds to the precision of the servo, commonly less than one degree. Since the servo has a maximum rotational capability (around 210), the camera is initially rotated to its leftmost position so that in stage 2 it can scan through the entire angular range of the servo.

5 Conclusions and Future Work

The MRSCL environment opens new perspectives in the scope of interactive educational robots. It takes advantage of handheld mobility and portability to support face-to-face collaborative work. In particular, the wireless network used in MRSCL Geometry allows students to explore the problem through different physical points of view, developing their spatial thinking abilities. Since each student has control over just one part of the solution, collaboration and discussion emerges naturally, providing a dynamic window into the mathematical meanings in construction. Finally, MRSCL Geometry is a tool to geometrically solve a geometrical problem, abolishing the algebraization flaw in mathematical teaching and allowing students to develop their geometrical abilities and visual reasoning. Due to the simplicity of the interface and the reduced set of activity rules, not much training is required by the students to use the application, diminishing the overhead introduced by this new technology. In relation to robotic vision, real time landmark detection and tracking are required. Due to limitations in the camera transmitting bandwidth, algorithms based on binary images were developed for this purpose. These algorithms, based on well-chosen landmarks, prove to be efficient and robust, accomplishing all the real time requirements of the activity.

Our future work focuses on the experimental validation of the activity. We want to validate the fact that students using MRSCL Geometry attain a better learning and understanding of geometry, measurement, and estimation, than students who do not use this technology. To do this, we will follow a methodology similar to the one used in our previous research [12], using one experimental and two control groups. Throughout the experiment, the experimental group will work with MRSCL Geometry while control group 1 will work with a similar collaborative activity implemented without technology. Control group 2 will remain unaltered from its regular class activities. Students will be examined through a Pre-Test, Post-Test, and delayed Post-Test. In this way, we will be able to determine whether the learning differences, if any, arise from the collaborative activity itself, the MRSCL technological improvement or both. The delayed Post-Test will also retrieve information about how deep-rooted where the learned concepts.

References

1. R. Avanzato and P. S. Abington. Mobile robot navigation contest for undergraduate design and k-12 outreach. In *ASEE Annual Conference Proceedings*. American Society for Engineering Education, 2002.

2. D. H. Clements and M. T. Battista. Constructivist learning and teaching. *Arithmetic Teacher,* 38:34–35, 1990.
3. D.H. Clements. Teaching length measurement: Research challenges. *School Science and Mathematics,* 99(1):5–11, 1999.
4. C. Cortez, M. Nussbaum, R. Santelices, P. Rodriguez, and G. Zurita. Teaching science with mobile computer supported collaborative learning (MCSCL). In *Second IEEE International Workshop on Wireless and Mobile Technologies In Education (WMTE 2004),* pages 67–74. IEEE Computer Society Press, 2004.
5. C. Hoyles. Modelling geometrical knowledge: The case of the student. In *Intelligent Learning Environments: The Case of Geometry,* volume 117, pages 94–112. Springer-Verlag, 1996.
6. C. Hoyles and R. Noss. What can digital technologies take from and bring to research in mathematics education? In *Second International Handbook of Mathematics Education,* pages 323–349. Kluwer Academic Publishers, 2003.
7. D. Johnson and R. Johnson. *Learning Together and Alone.* Allyn & Bacon, 5th edition, 1999.
8. J. Piaget and B. Inhelder. *The child's conception of space.* W. W. Norton, 1967.
9. L. Vigotsky. *El desarrollo de los procesos psicolgicos superiores.* Crítica, Barcelona, 1979.
10. E. Wang and R. Wang. Using legos and robolab (labview) with elementary school children. In *Frontiers in Education Conference,* volume 1, page T2E/11. IEEE, 2001.
11. J. Weinberg, G. Engel, K. Gu, C. Karacal, S. Smith, W. White, and X. Yu. A multidisciplinary model for using robotics in engineering education. In *ASEE Annual Conference Proceedings.* American Society for Engineering Education, 2001.
12. G. Zurita and M. Nussbaum. A constructivist mobile learning environment supported by a wireless handheld network. *Journal of Computer Assisted Learning.* Accepted for publication.
13. G. Zurita and M. Nussbaum. Computer supported collaborative learning using wirelessly interconnected handheld computers. *Computers & Education,* 42(3):289–314, 2004.

Evaluation of the Teaching-Learning Process with Fuzzy Cognitive Maps

Ana Lilia Laureano-Cruces[1] , Javier Ramírez-Rodríguez[1], and Amador Terán-Gilmore[2]

Universidad Autónoma Metropolitana – Azcapotzalco
[1] Departamento de Sistemas
[2] Departamento de Materiales
Av. San Pablo 180, Col. Reynosa Tamps
{clc, jararo, tga}@correo.azc.uam.mx

Abstract. The evaluation of the teaching-learning process is a conduct that poses great challenges from a modeling perspective. Tutor modules form part of intelligent systems that are applied to education, and should assess: when to interrupt, how to teach a given topic, and what to teach in a given moment. By considering the different types of errors, a good evaluation of the teaching-learning process allows the systems to adapt in an optimum form to different users. This paper proposes a design technique to model the conduct of an expert that evaluates the results of the teaching-learning process within a reactive learning environment. To achieve this, a fuzzy cognitive map, which is a representation recently proposed to model the conduct and operation of complex systems, is used. Bart Kosko introduced this type of maps in 1986 with the aim of describing the behavior of a system in terms of concepts and causal relations between concepts. Since then, their use has been extended to diverse real-world situations that span from the analysis of investment in stock to supervisory system control. Missing conceptions, which provide a basis to assemble the didactic tactics, are imputed to this representation. Starting from the mental model of the expert in the learning domain, a conceptual genetic graph is established to orientate the fuzzy cognitive map. Finally, this paper presents a concrete example.

Keywords: fuzzy cognitive maps, genetic graph, cognitive task analysis, mental models, intelligent learning systems, reactive learning environment, qualitative process.

1 Introduction

According to Estévez [7], we tackle deep problems when we ask ourselves: 1) Are students really learning?; 2) What is the degree of applicability of their acquired knowledge within the environment for which they are being prepared?; 3) Do they have enough knowledge to continue studying in colleges and universities?; 4) And due to the multidisciplinary interaction required by the current pace of scientific and technological development, When does the acquired knowledge becomes obsolete?

Facing such perspective, we pretend to enrich the didactic design with cognitive sciences, in order to introduce these kinds of processes into teaching and learning through the design and use of cognitive strategies. Therefore, this proposal is based on: 1) A multi-nodal perspective (based on different theoretical sources); 2) A holistic and complete approach (knowledge, abilities, attitudes, values); 3) Using the cognitive components from the expert and the learner; 4) Prioritizing the use of cognitive strategies as means to activate the mental processes needed for the learning process. In the last case, the benefit is owing to compare the learner's development to the expert's development considering both mental models as a guide of the teaching - learning process. The expert's mental model is considered as a model produced from his/her scientific knowledge of the integrated domain. This conceptual frame is what allows to: think, analyze and take decisions.

The cognitive components are: 1) in the case of the learner's domain: declarative knowledge, strategies, tactics, and mental schemes, 2) in the case of the expert's domain: the developed procedures of pattern recognition, the chained actions procedures , the solutions' search strategies improvements, the declarative knowledge structure and finally the mental models.

The final result of a didactic enriched design that is known as a cognitive model, and it is the final representation of an integrated set of strategic components that allows to: 1) Sequence the material, 2) Use conceptual graphs, 3) Use examples, 4) Incorporate practical experience in a given moment, and 5) Use strategies to motivate the students. Another important aspect of this didactic model is that it should exhibit the different aspects involved during learning, with the purpose of achieving the objectives in the best possible way and under anticipated conditions.

In this kind of models the learner's determines the cognitive approach behavior perception and understanding of the situation related to his/her goals or purpose and that learning is a change in perception and understanding rather than change in behavior. For the development of this kind of cognitive models there exists a methodology proposed by de Arriaga et al. [2] to take into account the transfer from novices into experts in the teaching-learning process design. The elements used during the development of this work are introduced next.

2 Human Behavior Domain: Evaluation of the Teaching-Learning Process

2.1 Cognitive Sciences

Cognitive sciences are those that study the human mind from the perspective of a system that receives, stores, recovers, transforms and transmits information, with the goal of learning and problem solving. Cognitive sciences, originated in the 50s, arise as an attempt to make the common interests of cognitive psychologists, researchers in artificial intelligence, linguists and philosophers, converge. Cognitive sciences have the goal of understanding how the human mind functions in terms of information processing. According to Estévez [7], one of the main concepts that have revolutionized the inclusion of cognitive sciences in the development of didactic models is the conception of knowledge as an intern representation that is built and organized in internal structures known as mental schemes or models [6,15 and 18].

Mental schemes allow the knowledge of the different states that in turn allow the maturity of expertise and the behavioral differentiation of novices and experts during problem solving that imply the use of different strategies to arrive to the solution. In this sense, they are an element to be considered during the development of software systems that use artificial intelligence techniques, such as intelligent learning systems and expert systems, among others.

2.2 Mental Schemes or Models

In some cognitive areas, it is possible to formulate competence theories that specify what has to be calculated. When and why; and later, to develop a representative algorithm based on these theories. This study area is known as competence theory and is carried out through mental schemes.

An objective of mental models is to establish relationships between the qualitative models and causal explanations, which allow the student to get involved in different learning strategies such as exploration, and requesting tutorial demonstrations, explanations and problem solving. These relationships allow the use of different types of reasoning with the purpose of accessing the different scenarios created to facilitate the learning of the concepts involved in the specific knowledge of the domain (from a qualitative point of view) [20,21]. In our case of study this is referred in section three, with the experimental planning steps.

2.3 Genetic Graph

A genetic graph (GG) is a tool, developed by Goldstein [12,13] and based on the genetic epistemology of Jean Piaget [12], used to represent knowledge. A GG shows knowledge (of any type) grouped into nodes and the connectors that relate them. The connectors may imply order or inclusion, as in the case of Gagne's nested hierarchies [10]. The history and learning style of a student can also be recorded by taking into consideration the nodes that have been visited and the predilection of the connectors that have been used during the learning process.

The connectors in this graph can and have been broadened according to the needs of the domain to be modeled [3, 8, 18]. Next, an example of the type of connectors used in this graphs are offered: **PreCond** implies precedence (*before than*); **PostCond** implies subsequence, a knowledge that can only be accessed *after* covering the knowledge to which it is connected; **Anlg** implies the existence of correspondence (analogy) of the constants between nodes; **Class** assumes the existence of a conceptual or ability hierarchy; **SubClass** demands the existence of granularity levels in the definition of conceptual or ability abstractions; **ItIs** represents the definition of a specific component according to the domain in question. The connectors between nodes, besides indicating execution order for the different tasks, define the data and relationships that exist between the input and output of the nodes and the different abstraction levels (which may or may not exist) that represent the relationship between concepts, and offer a guideline to assemble the teaching-learning process. This is also related to obtaining the critical points of the general strategy and the competency of the different levels (in teaching terms). In our case of study this is refereed in section three, with Fig 2 and Table 1.

2.4 Fuzzy Cognitve Maps

Fuzzy congnitive maps (FCMs), which constitute a new approach to model the behavior and operation of complex systems, were introduced by Bart Kosko in 1986 [16,17] to describe the behavior of a system in terms of concepts and causal relationships between concepts. Since then, their use has been extended to diverse real-world situations that span from the analysis of investment in stock to supervisory system control, [1,5,24-31].

FCMs are represented through a digraph, in which the nodes are concepts that describe the system's main characteristics, and the edges between nodes establish causal relationships (positive or negative) between concepts. This graphical representation (Fig. 1) illustrates the influence that each concept has on the rest. Next, a FCM will be graphically illustrated. The concepts in an FCM are events, whose values change in time, and that originate in the system. Concepts take values within the interval [0,1]; and the interconnection weights, p_{ij}, within the interval [-1,1]. As shown in the graph, the value corresponding to the edge located between concepts *i* and *j* can be represented through p_{ij}. $p_{ij} = 0$ indicates the absence of a relation between concepts *i* and *j*. $p_{ij} > 0$ denotes a positive causality, which implies that an increment in concept *i* results in an increment in concept *j* or that a decrement in concept *i* results in a decrement in concept *j*. If $p_{ij} < 0$, a negative causality exists, implying an increment in concept *i* results in an decrement in concept *j* or that a decrement in concept *i* results in a increment in concept *j*.

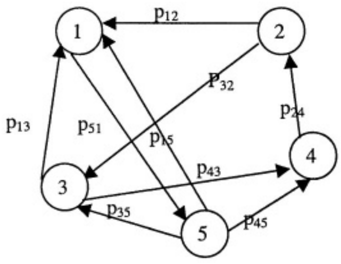


Fig. 1. Fuzzy Cognitive Map

A proper vertex selection is very important, because not only should the relevant concepts of the system be identified, but their activation modeled in such way that causal relationships are properly identified. An adjacency matrix, *P,* is used to represent the cause-effect relationships among nodes. The value of each concept at instant *t+1* is determined from the current value of the concept and the matrix *P.* One of the earliest manners in which it was used is as follows:

$$C_i(t+1) = f\left[\ \sum_{k=1}^{n} p_{ki}(t_n)C_k(t_n)\ \right] \qquad (1)$$

In [28], Stylios and Groumpos establish more refined formulations.*f* is a threshold function that is used to map the value of the concept within a normalized interval. Proper selection of *f* is also very important; [25] shows how the selection of *f* affects a system. In our case of study this is refereed in section three, with Fig 3 and Table 2.

3 Cognitive Analysis and Modeling

Within a reactive environment, the experimental planning established by the expert represents the mental model. The goal is that the user learns based in the qualitative reasoning theory [9] (cause-effect). To this end, a reactive environment is modeled [4]. This paper will not cover the learning domain; for further information the reader can refer to [32]. Starting from the experimental planning, the cognitive task analysis will be developed. The conceptual genetic graph, which indicates the relationships between concepts and provide guidelines to assemble the teaching-learning process, will be established from these two elements. The results that are discussed next form part of an incremental project into which results have already been incorporated [19-22].

Starting from the experimental planning, a concrete example is developed next. Then, the cognitive task analysis (CTA) is developed [23]. Once the CTA has been developed (Table 1), the conceptual genetic graph is established (Fig. 2). The graph includes three key concepts for the formulation of the equation of motion of a single-degree-of-freedom system, main learning objective of the reactive environment.

The equation of motion of a single-degree-of-freedom system can be established through the following steps:

a) Define if the structure is a single-degree-of-freedom system. Within the context of this tutor, this implies checking if it satisfies geometric and motion restrictions imposed by physical and cinematic conditions.
b) Establish clearly the degree of freedom (displacement or rotation) that will be used to formulate the equation of motion.
c) Establish, from the actual mass distribution in the structure, the value of the lumped mass corresponding to a mass-spring-dashpot system.
d) Establish, from the structural elements of the structure, the stiffness value of the spring of the mass-spring-dashpot system. The stiffness of the spring is actually the force that should be applied to the mass to induce in it a unitary displacement in the direction of the degree of freedom established in b).
e) Establish the damping coefficient of the dashpot. The tutor will provide this coefficient.
f) Establish the equation of motion of the simple structure from the properties of the mass-spring-dashpot system.

Basically, in this example there are two connectors which are not complementary, and that imply access order to the abilities: 1 and 2. Although connector type *"before than"* (**PreCond**) implies an ability that has to be dominated before entering another node (pre-requisite), it does not involve order. Nevertheless, when connector type *"after than"* (**PostCond**) is included, an access order is implied.

Table 1. Cognitive task analysis of the experimental planning

Step	Conceptual knowledge	Evaluation type	Representation type	Abilities
a	1. Establish if the mass of the structure can be modeled as lumped 2. Motion restricted to a plane (translation or rotation)	Graphical examples and evaluation of graphical examples	Process	Pattern recognition
b	Mass motion	✓	Process	Strategic
c	Distinction between mass and weight and quantification of weight	✓	Process	Procedural
d	1. Identify all structural elements that provide stiffness 2. Correct computation	✓	Process	Procedural
e	Given	Given	Structures	---
f	Equation of motion	Known pattern	Structures	Procedural

In the case of reactive environments, the system has mechanisms that immediately respond to the student actions. The system reacts to the request of evaluating the hypothesis related to the measures the student has based in the parameters that are involved. Because of this, the teaching-learning control will be assembled taking into consideration the pre and post condition connectors. The conceptual gentic graph (Fig. 2) implies the student missing conception (it is represented by the input number 9 in Table 2).

In each step there are errors that can be made, and thanks to the expert, the origin of the error is known. The environment, which is represented in this case by the state of the reactive interface, will be constantly surveyed. The state will have a symbolic representation described by the presence or absence of certain concepts, necessary in each one of these steps. Once the teaching-learning process control is applied, the output, which is represented by a binary chain, is interpreted (Table 2). These outputs (binary chain) will be connected to a series of didactic actions.

Next, an example is presented. Fig. 3 shows the FCM proposed by the expert, and a set of possible inputs that represent the environment state and its corresponding outputs, whose interpretation sends to the different didactic tactics.

In our study case we have simplified the fuzzy rank of the inputs to the FCM, being the choice zero or one, for our first tests, however, these can be assigned according to the mistake's seriousness in a rank between [0,1].

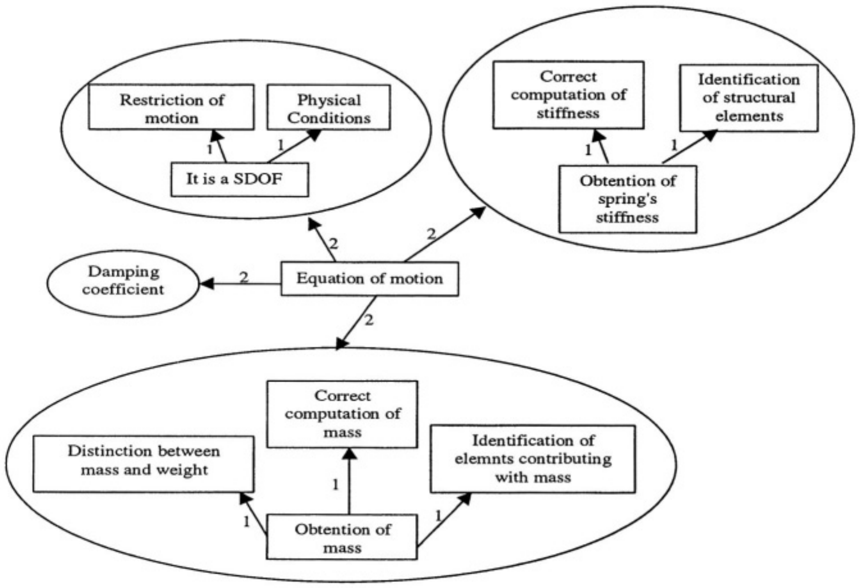

Restriction of motion
Physical Conditions
It is a SDOF
1
1
Correct computation of stiffness
Identification of structural elements
Obtention of spring's stiffness
1
1
2
2
Damping coefficient
2
Equation of motion
2
Correct computation of mass
Distinction between mass and weight
Identification of elemnts contributing with mass
1
1
1
Obtention of mass

Fig. 2. Conceptual genetic graph

Based on the work of García *et al.* [11], the fuzzy cognitive map shown in Fig. 3 was elaborated. This figure shows the events related to the teaching - learning process. In this case the node labelled as ERROR directly feeds the NEED FOR INTERRUPTION, the node labelled IDLE TIME directly feeds THE USE OF INCENTIVES. Table 2 shows two different situations derived from two different initial states.

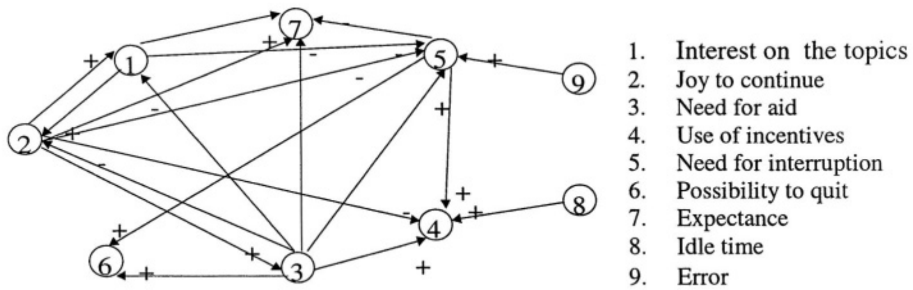


Fig. 3. Fuzzy Cognitve Map

The different conditions of the FCM stabilisation, due to different actual events in a specific moment of the teaching-learning process, represent the events that the tutor will generate through didactic actions with the purpose of having a reliable teaching - learning process. The different didactic tactics will be assembled according to the event that gave rise to that condition and to the conceptual mistake. For instance, on the example 1, they are presented in an initial condition: lack of interest (1) and lack of enthusiasm (2) on the topic, and obviously, lack of expectancy (learning stage, 7). On the first stage, a need of help direct from the tutor and the usage of incentives are suggested. On the second stage, it is suggested to make a direct interruption (for an explanation or to ask directly what it is not understood). On the third stage, it is suggested to use incentives. It is important to make a stand on the fact that the possibility to quit was present on every stage; and on the fact that after the second stage it arrives the presence of expectancy.

Table 2. Fuzzy cognitive map simulation

Partial states	1	2	3	4	5	6	7	8	9	-	1	2	3	4	5	6	7	8	9
Initial state	-1	-1	0	0	0	0	-1	0	0	-	0	0	0	0	0	0	-1	1	1
First state	0	0	1	1	0	1	0	0	0	-	0	0	1	1	0	1	0	0	0
Second state	0	0	0	1	1	1	0	0	0	-	0	0	0	1	1	1	0	0	0
Third state	0	0	0	1	0	1	1	0	0	-	0	0	0	1	0	1	1	0	0
Final state	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0

The tutor will choose the posed chain of actions or just one of them. We should remember that the learner's mental model constantly changes and it is pretended that every didactic action has a beneficent effect on the learning process.

On the other side, the didactic tactics that the tutor may use are represented by the set {3,4,5} of the FCM nodes. This is the reason why the tutor's conductual pattern is around these didactics' tactics. It is expected to have a more specific action when deeping on the mistake's causes (Fig. 2) and depending on the mistake's seriousness the weight assigned to the FCM input will be a fuzzy rank between [0,1]. All these will allow a richer topology.

4 Conclusions

The following conclusions can be reached from the results that have been obtained: a) A proposal to control through FCMs the diagnostic process in an intelligent learning system is presented; b) An advantage of this is the ability to include the expert's knowledge by taking advantage of the conceptual genetic graph; thus avoiding the symbolic representation of behavioral reasoning based in rules; c) An indispensable characteristic of reactive environments is the need to continuously perceive and interpret its changes. This is because learning is evaluated based on the theory of qualitative reasoning in which the cause-effect actions are the base mechanisms of the reactive environments. The FCM allow a faster control of the different states of the environment.

Acknowledgment. This work was partly financed by Consejo Nacional de Ciencia y Tecnología (SEP-CONACYT) under grant number 39970 - H.

References

1. Aguilar, J "A Dynamic Fuzzy-Cognitive-Map Approach Based on Random Neural Networks", *International Journal of Computational Cognition (http://YangSky.com/yangijcc.htm),* Vol. 1, No. 4, pp. 91-107, December 2003.
2. de Arriaga, F., El Alami, M., Ugena, A. Acceleration of the Transfer of Novices into Experts: the Problem of Decision Making, *Proceedings International Conference BITE'0 ,* University of Eindhoven, 2001, 245-258.
3. Bretch, B. and Jones, M. Student Models: The Genetic Graph Approach. *Man Machine Studies.* Vol. 28, 1988, 483-504.
4. Brown, J. S. Burton, R.R. Bell, A. G. SOPHIE, a Step Towards a Reactive Learning Environment. *Int. J. Man Machine Studies.* Vol. 7, 675-696,
5. Carlsson, C. Fullér, R. Adaptive Fuzzy Cognitive Maps for Hyperknowledge Representation in Strategy Formation Process, *Proceedings of International Panel Conference on Soft and Intelligent Computing,* Technical University of Budapest, 1996, 43-50, (http://www.aboifil/~fuller/asic96.pdf).
6. Elio, R. Scharf, P. Modeling Novice-to-Expert Shifts in Problem-Solving Strategy and Knowledge Organization. *Cognitive Science* 14, 1990, 579-639.
7. Estévez-Nénninger, E.H. *Enseñar a Aprender: estrategias cognitivas.* Colección Maestros y Enseñanza. (Ed.) Piados. México-Buenos Aires-Barcelona, 2002.
8. Fernández, I. Estrategias de Enseñanza en un Sistema Inteligente de Enseñanza Asistida por Ordenador. *Tesis Doctoral* (Tercer Ciclo) de la Universidad del País Vasco, San Sebastián 1989.
9. Forbus, K. .D. Qualitative Process Theory. *Journal Artificial Intelligence* 24: 85-168. 1984.
10. Gagné, E. *The Cognitive Psychology of school Learning.* Boston, MA: Little Brown & Company. 1985.
11. García, H., Reyes, C. and Morales, R Diseño e implementatión de Mapas cognitivos difusos para tutoriales inteligentes. *Memorias del XV Congreso Nacional y I Congreso Internacional de Informática y Computación de la ANIEI.* Vol. I, octubre, 2002, 51-59.
12. Ginsburg, H. Opper,, S. *Piaget y la Teoría del Desarrollo Intelectual.* Prentice Hall. 1986.
13. Goldstein, I. The Computer as a Coach: An Athletic Paradigm for Intellectual Education. *Memo no. 389,* December 1976, *MITArtificial Intelligence Laboratory.*
14. Goldstein, I. The Genetic Graph: A Representation for the Evolution of Procedural Knowledge. *Man Machine Studies.* Vol. 11, 1977, 51-77.
15. Johnson-Laird, P. N. *Mental Models.* Cambridge University Press. Cambridge, Mass.:Harvard University Press. 1983.
16. Kosko, B. Fuzzy Cognitive Maps, *International Journal ofMan-Machine Studies,* Vol. 24, 1986, 65-75.
17. Kosko, B. *Neural Networks and Fuzzy Systems. A Dynamical Systems Approach to Machine Intelligence,* Prentice-Hall, New Jersey, 1992.
18. Laureano, A. de Arriaga, F. El Análisis Cognitivo de Tareas: Una herramienta para modelar la conducta de los Sistemas de Enseñanza Inteligentes. *Revista Latina, de Pensamiento y Lenguaje,* Número Monográfico, 2B, Vol. 4. 1988, 315 - 335.
19. Laureano Cruces, A. L. Terán Gilmore, A. de Arriaga, F. A Learning Model Based on a Didactic Cognitive Approach: The Case Of Single-Degree-Of-Freedom Systems, *ComputerApplications in Engineering Education,* to appear, 2004.

20. Laureano Cruces, A. L. Terán Gilmore, A. de Arriaga, F. El Alami, M. La Importancia de Las Estrategias Cognitivas en el Diseño del Curricula Didáctico, *Memorias del XVI Congreso Nacional y II Congreso Internacional de Informática y Computación de la ANIEI.* Vol. I, Zacatecas, 22-24 de octubre del 2003, 35 – 41.
21. Laureano Cruces, A. L. Terán Gilmore, A. de Arriaga, F., Un Enfoque Didáctico-Cognitivo del Análisis de los Conceptos de los Sistemas de un Grado de Libertad, *Revista Digital Universitaria.* http://www.revista.unam.mx/ , Vol. 4, Num 7, 2003.
22. Laureano-Cruces, A. L. Ramírez-Rodríguez, J. de Arriaga-Gómez, F. Los Agentes Reactivos y la lógica Borrosa: Herramientas para Modelar el Entorno de los Sistemas de Enseñanza Inteligentes. *Memorias del CISCI 2002. Conferencia Iberoamericana en SISTEMAS, CIBERNÉTICA E INFORMÁTICA.* Vol. I, Orlando, International Institute of Informatics and Systems, Florida, EE.UU. July, 2002, 356-361.
23. Laureano, A. L. de Arriaga, F. García-Alegre, M. Cognitive task analisys: a proposal to model reactive behaviours. *Journal ofExperimental & Theoretical Artificial Intelliegence.* (13)(2001)227-239.
24. Miao, Y. Liu, Z. Q. On Causal Inference in Fuzzy Cognitive Maps, *IEEE Transactions on Fuzzy Systems,* Vol. 8, No. 1, 2000, 107-119.
25. Mohr, S. T. The Use and Interpretation of Fuzzy Cognitive Maps, *Master's Project,* Rensselaer Polytechnic Institute, http://www.voicenet.com/~mohr/fcm_white.html.
26. Schneider, M. Schnaider, E. Kandel, A. Chew, G. Automatic construction of FCMs. *Fuzzy Sets and Systems 93* (1998) 161-172.
27. Siraj, A. Bridges, S. M. Vaughn, R. B. Fuzzy Cognitive Maps for Decision Support in an Intelligent Intrusion Detection System, http://www.cs.msstate.edu /~bridges/papers/nafips2001.pdf.
28. Stylios, C. D. Groumpos.,P. P. Fuzzy Cognitve Maps: a model for intelligent supervisory control systems, *Computers in Industry* 39 (1999) 229-238.
29. Stylios, C. D. Groumpos.,P. P. Fuzzy Cognitve Maps: a soft computing technique for intelligent control, pp. 97-102. *Proceedings of the 15th IEEE International Symposium on Intelligent Control* (ISIC 2000).
30. Stylios, C. D. Groumpos.,P. P. Modeling Complex Systems Using Fuzzy Cognitive Maps, *IEEE Transactions on Systems, Man and Cybernetics. Part A: Systems and Humans,* Vol. 34, No. 1 (2004), 155-162.
31. Stylios, C. D. Groumpos.,P. P. The Challenge of modeling Supervisory Systems Using Fuzzy Cognitive Maps, *Journal of Intelligent Manufacturing 9* (1998), 339-345.
32. Terán-Gilmore, A. Diseño por Desempeño: Antecedentes, Conceptos Generales y Perspectivas, *Proceedings del VII Simposio Nacional de Ingeniería Sísmica* (CD). Cuernavaca, México. 29 - 30, November 2002.
33. Wild, M. Mental models and computer modelling. *Journal of Computer Assisted Learning.* Vol. 12, 1996, 10-21.

Using Simulated Annealing for Discrete Optimal Control Systems Design

Horacio Martínez-Alfaro and Martín A. Ruiz-Cruz

Center for Intelligent Systems, Tecnológico de Monterrey,
Monterrey, N.L. 64849 México
hma@itesm.mx

Abstract. This work presents an approach to solve the Discrete-Time Time Invariant Linear Quadratic Optimal Control problem which minimizes a performance index (either minimum time and/or minimum energy). The design approach presented in this paper transforms the LQ problem into a combinatorial optimization problem and use the Simulated Annealing algorithm to perform the optimization part. In addition, a non-conventional performance index is proposed. Some SISO and MIMO systems are tested with this approach. Their results proved the approach to be an excellent option to find a solution to the problem.

1 Introduction

A lot of research has been done in Automatic Control Systems during the last decade and more recently in discrete control systems due to the popular use of powerful personal computers. This work presents an approach to solve the Discrete-Time Time Invariant Linear Quadratic (LQ) Optimal Control problem which minimizes a specific performance index (either minimum time and/or minimum energy). The design approach presented in this paper transforms the LQ problem into a combinatorial optimization problem. The Simulated Annealing (SA) algorithm is used to carry out the optimization.

Simulated Annealing is basically an interactive improvement strategy augmented by a criterion for occasionally accepting configurations with higher values of the performance index [3,4,5,9]. Given a performance index $J(z)$ (analog to the energy of the material) and an initial configuration z_0, the iterative improvement solution is seeked by randomly perturbing z_0. The Metropolis algorithm [4,5,9] was used for acceptance/rejection of the perturbed configuration.

In our design approach, SA was used to minimized the performance index of the LQ problem and as result we obtain the values of the feedback gain matrix $\mathbf{K}$ that make stable the feedback system and minimizes the performance index of the control system in state space representation [7]. The SA algorithm starts with an initial $\mathbf{K}$ feedback gain matrix and evaluates the performance index. The current $\mathbf{K}$ is perturbed to generate another $\mathbf{K}_{new}$ and the performance index is evaluated. The acceptance/rejection criteria is based on the Metropolis algorithm. This procedure is repeated under a cooling schedule. Some experiments were performed with first through third order plants for Regulation and Tracking, SISO and MIMO systems. Matlab and Simulink were the simulation tools to carry out the experiments.

The parameters of the SA algorithm (perturbation size, initial temperature, number of Markov chains, etc.) were specially tunned for each plant.

Additional experiments were performed with non-conventional performance indices for tracking problems [8] where characteristics like maximum overshoot $\max(y(k) - r(k))$, manipulation softness index $|\mathbf{u}(k+1) - \mathbf{u}(k)|$, output softness index $|\mathbf{y}(k+1) - \mathbf{y}(k)|$, and the error magnitude $|\mathbf{r}(k) - \mathbf{y}(k)|$.

Our proposed scheme with the use of the SA algorithm showed to be another good tool for discrete optimal control systems design even though only linear time invariant plants were considered [1,2,6,7,10]. A large CPU time involved in our scheme in order to obtain similar results to LQ. The design process is simplified due to the use of gain matrices that generate a stable feedback system. The equation required are those use for the simulation of the feedback system which are very simple and very easy to implement.

2 Methodology

The procedure is described as follow:

1. Propose a initial solution $\mathbf{K}_{initial}$.
2. Evaluate the performance index and save initial cost $J_{initial}(\mathbf{K}_{initial})$. $\mathbf{K}_{initial}$ needs to be converted to matrices $\mathbf{K}_1$ and $\mathbf{K}_2$ for tracking systems.
3. Randomly perturb $\mathbf{K}_{initial}$ to obtain a $\mathbf{K}_{new}$.
4. Evaluate the performance index and save initial cost $J_{new}(\mathbf{K}_{new})$.
5. Accept or reject $\mathbf{K}_{new}$ according to the Metropolis criterion.
6. If accepted, $\mathbf{K}_{initial} \leftarrow \mathbf{K}_{new}$, decrement temperature according to $J_{new}/J_{initial}$.
7. Repeat from step 3.

Once a Markov chain is completed, decrement the temperature, $T_{i+1} = \alpha T_i$, where T_i represent the current temperature and $\alpha = 0.9$ [4]. The procedure ends when the final temperature or a certain number of Markov chains has been reached.

3 Implementation

The code was implemented in Matlab, and the models were design for Regulation and Tracking, SISO and MIMO systems.

A discrete optimal control system can be represented as follows [7]:

$$\mathbf{x}(k+1) = \mathbf{G}\mathbf{x}(k) + \mathbf{H}\mathbf{u}(k) \tag{1}$$

$$\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) + \mathbf{D}\mathbf{u}(k) \tag{2}$$

where $\mathbf{x}^{(n\times 1)}$ state vector, $\mathbf{y}^{(m\times 1)}$ output vector, $\mathbf{u}^{(r\times 1)}$ control vector, $\mathbf{G}^{(n\times n)}$ state matrix, $\mathbf{H}^{(n\times r)}$ input matrix, $\mathbf{C}^{(m\times n)}$ output matrix, and $\mathbf{D}^{(m\times r)}$ direct transmission matrix.

In an LQ problem the solution determines the optimal control sequence for $\mathbf{u}(k)$ that minimizes the performance index [7].

3.1 Regulation

The equation that define the performance index for a Regulator is [6]:

$$J = \frac{1}{2}\sum_{k=0}^{N-1}[\mathbf{x}'(k)\mathbf{Q}\mathbf{x}(k) + \mathbf{u}'(k)\mathbf{R}\mathbf{u}(k)] \tag{3}$$

where $\mathbf{Q}^{(n\times n)}$ positive definite or positive semidefinite Hermitian matrix, $\mathbf{Q}^{(n\times n)}$ positive definite or positive semidefinite Hermitian matrix, and N number of samples. Equation 3 represents the objective function of SA algorithm.

3.2 Tracking

A tracking system can be represented as follows [6]:

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{G}\mathbf{x}(k) + \mathbf{H}\mathbf{u}(k), & \mathbf{u}(k) &= \mathbf{K_1}v(k) - \mathbf{K_2}\mathbf{x}(k) \\ \mathbf{y}(k) &= \mathbf{C}\mathbf{x}(k), & \mathbf{v}(k) &= \mathbf{r}(k) - \mathbf{y}(k) + \mathbf{v}(k-1) \end{aligned} \tag{4}$$

where $\mathbf{x}$ state vector, $\mathbf{u}$ control vector, $\mathbf{y}$ output vector, $\mathbf{r}$ input reference vector, $\mathbf{v}$ speed vector, $\mathbf{K}_1$ integral control matrix, $\mathbf{K}_2$ feedback matrix, $\mathbf{G}$ state matrix, $\mathbf{H}$ input matrix, and $\mathbf{C}$ output matrix.

The representation used in this work was a Regulator representation [6]:

$$\xi(k+1) = \hat{\mathbf{G}}\xi(k) + \hat{\mathbf{H}}\mathbf{w}(k), \qquad \mathbf{w}(k) = -\hat{\mathbf{K}}\xi(k) \tag{5}$$

where:

$$\begin{aligned} \xi(k) &= \begin{bmatrix} \mathbf{x}_e(k) \\ \mathbf{u}_e(k) \end{bmatrix}, & \hat{\mathbf{G}} &= \begin{bmatrix} \mathbf{G} & \mathbf{H} \\ \mathbf{0} & 0 \end{bmatrix} \\ \hat{\mathbf{H}} &= \begin{bmatrix} \mathbf{0} \\ \mathbf{I}_m \end{bmatrix}, & \hat{\mathbf{K}} &= (\mathbf{R} + \hat{\mathbf{H}}'\hat{\mathbf{P}}\hat{\mathbf{H}})^{-1}\hat{\mathbf{H}}'\hat{\mathbf{P}}\hat{\mathbf{G}}, \\ [\mathbf{K}_2 \ \ \mathbf{K}_1] &= (\hat{\mathbf{K}} + [0 \ \ \mathbf{I}_m])\mathbf{R}\begin{bmatrix} \mathbf{G}-\mathbf{I}_n & \mathbf{H} \\ \mathbf{CG} & \mathbf{CH} \end{bmatrix}^{-1} \end{aligned} \tag{6}$$

and the states are defined as

$$\mathbf{x}_e(k) = \mathbf{x}(k) - \mathbf{x}(\infty), \qquad \mathbf{u}_e(k) = \mathbf{u}(k) - \mathbf{u}(\infty) \tag{7}$$

The performance index is:

$$J = \frac{1}{2}\sum_{k=0}^{\infty}[\xi'(k)\hat{\mathbf{Q}}\xi(k) + \mathbf{w}'(k)\mathbf{R}\mathbf{w}(k)] \quad \text{with} \quad \hat{\mathbf{Q}} = \begin{bmatrix} \mathbf{Q} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{8}$$

Since our simultaion is finite, the performance index should be evaluated for N samples:

$$J = \frac{1}{2}\sum_{k=0}^{N}[\xi'(k)\hat{\mathbf{Q}}\xi(k) + \mathbf{w}'(k)\mathbf{R}\mathbf{w}(k)] \tag{9}$$

Non-conventional Performance Index. Non-conventional performance indexes are specially good when we desire to include certain output and/or vector control characteristics in addition to the ones provided by a standard LQ problem.

The propose performance index is [8]:

$$J = C_1\zeta + C_2\vartheta + C_3\varphi + \sum_{k=0}^{N}[C_4\xi'(k)\hat{\mathbf{Q}}\xi(k) + C_5\omega'(k)\mathbf{R}\omega(k) + C_6\varepsilon(k)] \quad (10)$$

where

- ζ is the softness index of $\mathbf{u}(k)$ defined by $|\mathbf{u}(k+1) - \mathbf{u}(k)|$.
- ϑ is the maximum overshoot defined by $\max(\mathbf{y}(k) - \mathbf{r}(k))$.
- φ is the output softness index defined by $|\mathbf{y}(k+1) - \mathbf{y}(k)|$.
- $\varepsilon(k)$ is the error defined by $|\mathbf{r}(k) - \mathbf{y}(k)|$.
- $\xi(k)$ is the augmented state vector.
- $\mathbf{w}(k)$ is the vector under the control law.
- $\hat{\mathbf{Q}}$ and $\mathbf{R}$ are the weighting matrices for quadratic error.
- $C_i, i = 1, \ldots, 6$ are weighting constants. C_4 y C_5 take 0 or 1 values wheather or not to include the quadratic error.

This description is valid only for SISO systems. The changes for MIMO systems (we consider just n inputs and outputs) are:

- Softness index in vector $\mathbf{u}(k)$

$$\zeta = \max(\max(|u_i(k+1) - u_i(k)|), \ i = 1, \ldots, n) \quad (11)$$

- Maximum overshoot

$$\vartheta = \max(\max(y_i(k) - r_i(k)), \ i = 1, \ldots, n) \quad (12)$$

- Output softness index

$$\varphi = \max(\max(|y_i(k+1) - y_i(k)|), \ i = 1, \ldots, n) \quad (13)$$

- Error

$$\varepsilon(k) = \max(\max(|r_i(k) - y_i(k)|), \ i = 1, \ldots, n) \quad (14)$$

The SA algorithm is based on the one used in [4].

4 Experiments and Results

For SISO systems, many experiments were performed for regulator and tracking systems. In this work we present just the experiments with third order plants. Very similar experiments were performed with MIMO systems (regulator and tracking), but we only work with two-input-two-output plants.

4.1 SISO Systems

Regulator. The following values for a third order system were:

$$\mathbf{G} = \begin{bmatrix} 0 & 0 & -0.25 \\ 1 & 0 & 0 \\ 0 & 1 & 0.5 \end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \mathbf{Q} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{R} = 1, \quad \mathbf{x}(0) = \begin{bmatrix} -5 \\ 4.3 \\ -6.8 \end{bmatrix}$$

The SA algorithm parameters were: initial solution = **0**, maximum perturbation = 1, initial temperature = 100, number of Markov chains = 100, percentage of acceptance = 80. The SA algorithm found a J = 68.383218 and LQ a J = 68.367889. Table 1 shows the gains.

Table 1. Controller gains

	J	$\mathbf{K}$
LQ	$J = 68.367889$	$[-0.177028 \quad -0.298681 \quad -0.076100]$
SA	$J = 68.383218$	$[-0.193591 \quad -0.312924 \quad -0.014769]$

Figure 1(a), presents the SA behavior. The states of both controllers performs similarly, Figure 1(c); but we can appreciate that exist a little difference between them, Figure 1(b).

According to section 3.2, the tracking system experiment is next. The sample number is 100.

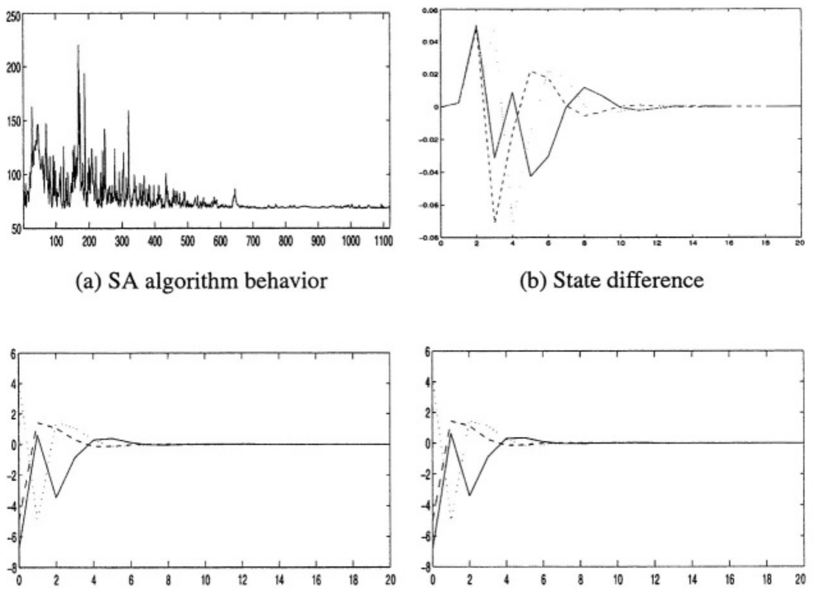
$$\mathbf{G} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -.12 & -.01 & 1 \end{bmatrix}, \mathbf{H} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \mathbf{C}^T = \begin{bmatrix} 0.5 \\ 1 \\ 0 \end{bmatrix}, \mathbf{Q} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{R} = 10$$

Yielding

$$\hat{\mathbf{G}} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -0.12 & -0.01 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \hat{\mathbf{H}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

The SA algorithm parameters were: initial solution = 0, maximum perturbation = 0.01, initial temperature = 100, number of Markov chains = 100, percentage of acceptance = 80. LQ obtained a J = 2.537522 and SA a J = 2.537810. Although the indexes are very similar, gain matrices differ a little bit (shown in Table 2). Figure 3 shows the states and the input.

Tracking with Non-conventional Performance Index. Several experiments were perform with type of index. This experiment was a third order plant, the same of previous section. The coefficient values for the performance index were: $C_1 = 10$, $C_2 = 10$,



(a) SA algorithm behavior (b) State difference

(c) States using LQ (d) States using SA

Fig. 1. Behavior of the SA algorith and the states using LQ and SA

Table 2. Controller gain

	$\mathbf{K}_1$	$\mathbf{K}_2$
LQ	0.290169	[−0.120000 0.063347 1.385170]
SA	0.294318	[−0.107662 0.052728 1.402107]

$C_3 = 20$, $C_4 = 1$, $C_5 = 1$, and $C_6 = 10$. The SA algorithm parameters were: initial solution = 0, maximum perturbation = 1, initial temperature = 100, number of Markov chains = 100, and the percentage of acceptance = 80. SA obtaained a J = 46.100502, with $\mathbf{K}_1 = 0.383241$, and $\mathbf{K}_2 = [-0.108121,\ 0.189388,\ 1.424966]$. Figure 4(a) shows the response of the system and Figure 4(b) show the states.

4.2 MIMO Systems

Regulator. The system used was:

$$\mathbf{G} = \begin{bmatrix} 3.5 & 0.5 & 0.5 \\ 1 & 2.5 & 0 \\ 1.5 & -1 & 4 \end{bmatrix}, \mathbf{H} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \mathbf{Q} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{R} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \mathbf{x}(0) = \begin{bmatrix} 5 \\ -1 \\ 3 \end{bmatrix}$$

The SA algorithm parameters were: initial solution = 0, maximum perturbation = 5, initial temperature = 100, number of Markov chains = 100, and percentage of

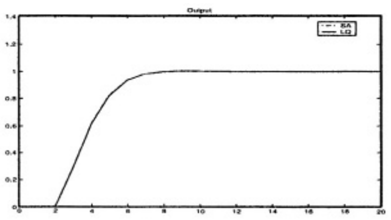


Fig.2. Output

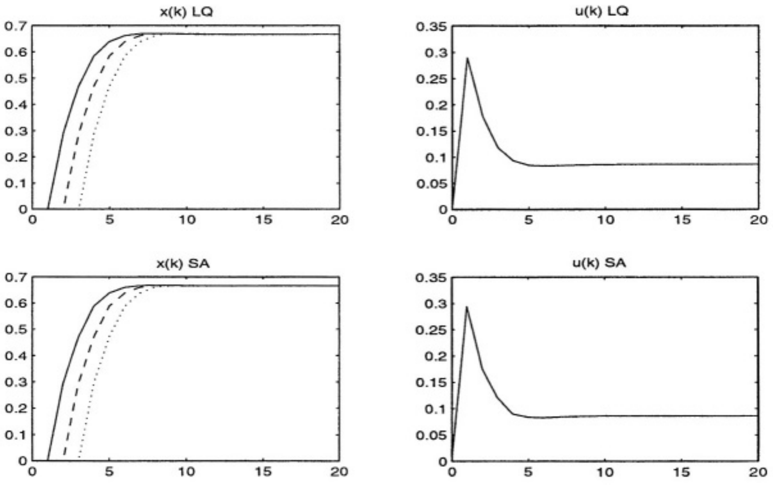


Fig. 3. States and input behavior

acceptance = 80. LQ obtained a J = 732.375702 and SA a J = 733.428460. Gain matrices are very similar. Figure 5 shows the states.

Tracking. The number of samples was 100.

$$\mathbf{G} = \begin{bmatrix} -\frac{1}{3} & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{2} \end{bmatrix}, \mathbf{H} = \begin{bmatrix} 2 & 3 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, \mathbf{C}^T = \begin{bmatrix} 4 & -1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, \mathbf{Q} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{R} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Converting the tracking system to regulator

$$\hat{\mathbf{G}} = \begin{bmatrix} -\frac{1}{3} & 0 & 0 & 2 & 3 \\ 0 & \frac{1}{2} & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \hat{\mathbf{H}} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

The SA algorithm parameters were: initial solution = **0**, maximum perturbation = 0.02, initial temperature = 100, number of Markov chains = 100, percentage of ac-

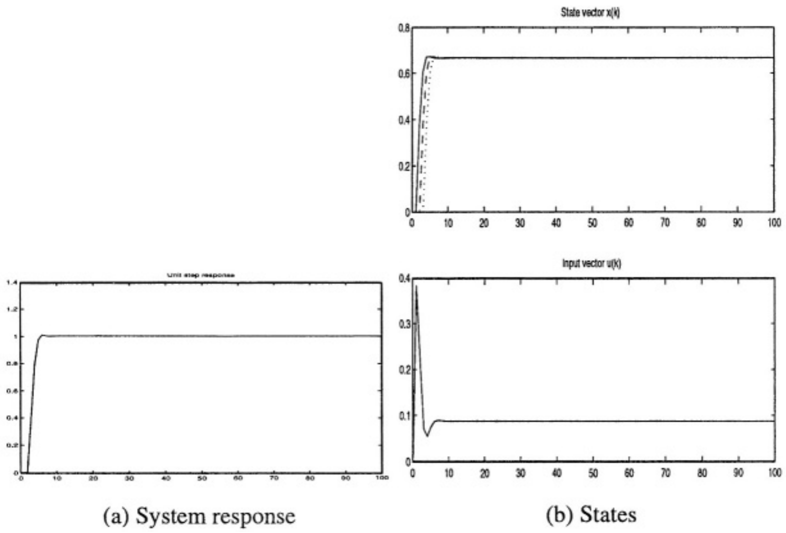


Fig. 4. Tracking with Non-conventional index

ceptance = 80. LQ obtained a J = 6.132915 and SA a J = 6.134467. The value entries obtained for the gain matrix differ a little bit from the ones obtained by SA; however, the performance indexes are very similar. Figure 6 shows the controller response.

Tracking with Non-conventional Performance Index. Although several experiments were performed, only one is shown here. The plant used for this experiment is the same as in the previous example and the performance index is the same as the tracking for SISO system example. The coefficient values were: $C_1 = 30$, $C_2 = 20$, $C_3 = 50$, $C_4 = 1, C_5 = 1$, and $C_6 = 30$. The SA algorithm parameters were: initial solution = **0**, maximum perturbation = 0.1, initial temperature = 100, number of Markov chains = 100, percentage of acceptance = 80. The results are:

$$J = 128.589993$$
$$\mathbf{K}_1 = \begin{bmatrix} -0.029799 & -0.874366 \\ 0.152552 & 0.560652 \end{bmatrix}$$

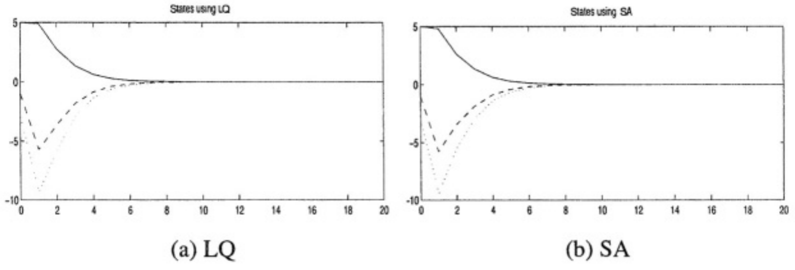


Fig. 5. MIMO Regulator system

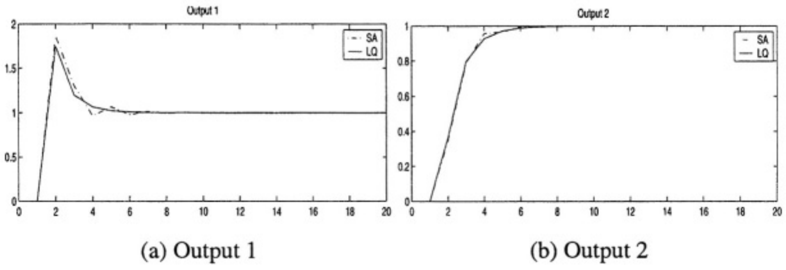


Fig. 6. MIMO tracking outputs

$$\mathbf{K}_2 = \begin{bmatrix} -0.279253 & 0.165710 & -0.398472 \\ -0.007700 & -0.132882 & 0.272621 \end{bmatrix}$$

The controller response is shown in Figure 7(a). The states are shown in Figure 7(b).

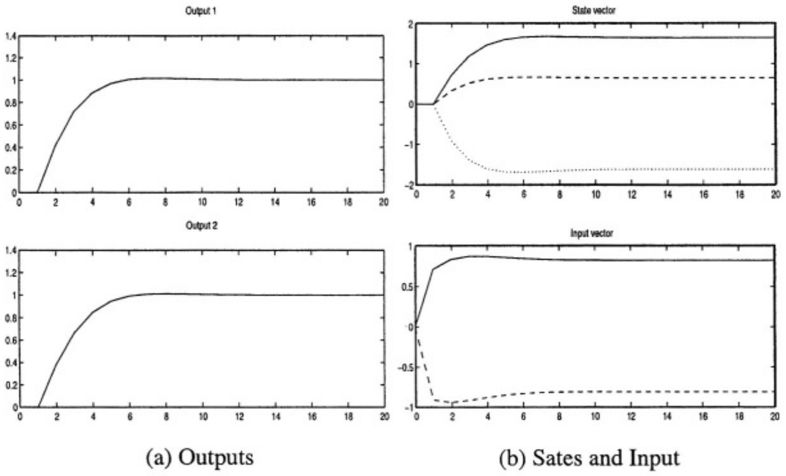


Fig. 7. MIMO tracking with non-conventional index

5 Conclusions

The results presented here, show that this kind of algorithms and the used technique, work well; however, it is not possible to generalize the employ of this method because we just consider first, second and third plants order. SA is an algorithm whose objective function most be adapted to the problem, and doing so (tuning), is where the employ of heuristics is required. Through these heuristics, we can propose the algorithm parameters values that make possible to find good solutions, but this is a tedious work. The CPU time that SA algorithm takes for finding a good solution is larger than the time we require to calculate LQ controller. But, in the case of tracking with non-conventional performance index, the method provided with SA algorithm works very well, and this is the main idea, to provide a good tool for discrete-time optimal control systems design.

Acknowledgments

The second author wishes to acknowledge the support in part for this research work granted by CONACYT, México under the project "Performance optimization in ad-hoc networks" and to the Center for Intelligent Systems, Tecnológico de Monterrey.

References

1. Mario E. Salgado Graham C. Goodwin, Stefan F. Graebe. *Control System Design.* Prentice-Hall, 2001.
2. Michael J. Grimble and Michael A. Johnson. *Optimal Control And Stochastic Estimation: Theory andApplications, Volume 1.* John Wiley and Sons, 1988.
3. A. Malhorta, J. H. Oliver, and W. Tu. Synthesis of spatially and intrinsically, constrained curves using simulated annealing. *ASME Advances in Design Automation,* DE(32):145–155, 1991.
4. H. Martínez-Alfaro and D.R. Flugrad. Collision-free path planning of an object using b-splines and simulated annealing. *in IEEE International Conference on Systems, Man, and Cybernetics,* 1994.
5. H. Martínez-Alfaro and Ulloa-Pérez. Computing near optimal paths in c-space using simulated annealing. *in ASME Design Engineering Technical Conference/Mechanisms Conference,* 1996.
6. Katsuhiko Ogata. *Discrete-time Control Systems.* Prentice-Hall, 1st. Edition, 1987.
7. Katsuhiko Ogata. *Sistemas de Control en Tiempo Discreto.* Pearson Educatión, Segunda Editcón, 1995.
8. Jaime Alberto Steffanoni Palacios. Diseño de sistemas de control óptimo en espacio de estados utilizando algoritmos genéticos. Master's thesis, ITESM, Campus Monterrey, 1998.
9. R. Rutenbar. Simulated annealing algorithms: An overview. *IEEE Circuit and Devices,* pages 19–26, Jan 1989.
10. Mohammed S. Santina, Allen R. Stubberud, and Gene H. Hostetter. *Digital Control System Design.* Sanders College Publishing, Second Edition, 1994.

Determination of Possible Minimal Conflict Sets Using Constraint Databases Technology and Clustering

M.T. Gómez-López, R. Ceballos, R. M. Gasca, and S. Pozo

Computer Engineering Superior Technical School of Seville, Spain
{mayte, ceballos, gasca, sergio}@lsi.us.es

Abstract. Model-based Diagnosis allows the identification of the parts which fail in a system. The models are based on the knowledge of the system to diagnose, and can be represented by constraints associated to components. Inputs and outputs of components are represented as variables of those constraints, and they can be observable and non-observable depending on the situation of sensors. In order to obtain the minimal diagnosis in a system, an important issue is to find out the possible minimal conflicts in an efficient way.

In this work, we propose a new approach to automate and to improve the determination of possible minimal conflict sets. This approach has two phases. In the first phase, we determine components clusters in the system in order to reduce drastically the number of contexts to consider. In the second phase, we construct a reduced context network with the possible minimal conflicts. In this phase we use Gröbner bases reduction. A novel logical architecture of Constraint Databases is used to store the model, the components clusters and possible minimal conflict sets. The necessary information in each phase is obtained by using a standard query language.

1 Introduction

Diagnosis allows to determine why a system correctly designed does not work as it was expected. It is based on the monitorization of a system. The diagnosis aim is to detect and to identify the reason of an unexpected behavior, or in other words, to identify the parts which fail in a system. Our proposal is based on DX [1] approaches and in other works as [2,3]. These works were proposed to find out the discrepancies between the observed and correct behaviors of the system.

In engineering applications it is often overlooked the storage of these data and query processing. The key idea in this paper is to combine the power of Constraint Databases (CDBs) [4,5] with the data treatment of diagnosis. We are able to improve the efficiency in some phases of the model-based diagnosis with CDBs. To improve the detection of possible minimal context conflicts, we use a program implemented in Java^{TM} and SQL (Standard Query Language). With CDBs technology [6] we are able to make the information and models persistent. Most DX approaches for components characterize the diagnosis of a system as a collection of minimal sets of failing components which explain the observed behaviors (symptoms). A conflict is a set of assumptions where, at least, one must be false. The assumptions are about behavioral modes of components. GDE [7] coupled with an ATMS [8] as inference engine uses previously discovered conflicts to constrain the search in the candidate space. The most important disadvantage of using this approach is the large number of possible conflicts ($2^n - 1$), n being the number of components.

In this work, we propose a new approach to automate and to improve the determination of possible conflicts, it is based on:

- A structural pretreatment in order to reduce drastically the computational complexity.
- The Reduction of the number of possible contexts to treat by means of symbolic techniques in order to obtain the possible conflicts.

This technique for elimination is Gröbner bases [9] that is our projection operator to manipulate multiple polynomial variables. It eliminates the non-observable variables of the constraints of the different contexts of a previous step.

Finding all minimal conflict has been a problem active enough at last years using CS-Tree [10]. Symbolic processing algorithms (Gröbner bases) of the initial model are used by model-based diagnosis communities [11,12]. Another proposition [13] presents the concept of a possible conflict as an alternative to the use of pre-compiled dependency-recording.

Our paper has been organized as follows: Section 2 reviews definitions and notation to allowing to formalize the subsequence operations. Section 3 shows an example to prove our solution. Section 4 describes the improvements to detect the possible minimal conflicts using CDBs and components clusters. Finally we present our conclusions and the future works in this research line.

2 Definitions and Notation

The definitions and notation used are based on the concepts proposed in the diagnosis community (DX). To introduce our work it is necessary the use of the following definitions and notation:

Definition 1. The System Polynomial Model (SPM): It can be defined as a finite set of polynomial equality constraints (P) which determine the system behavior. This is done by means of the relations between the non-observable variables (V_{nob}) and the observable variables (V_{ob}) which are directly obtained from sensors that are supposed to work correctly. Therefore, the tuple *SPM* (P, V_{ob}, V_{nob}) is obtained for a system.

Definition 2. Context Set (CS): A context set of a SPM is a collection of components which compose the system. The possible context set will be $2^{comp} - 1$, where *comp* is the number of components of the system.

Definition 3. Context Network (CN): A graph formed by all the elements of the context set of the system according to the way proposed by ATMS [8].

3 System Example: A System of Heat Exchangers

In order to explain our methodology, we will apply it to the system shown in Figure 1. It was presented in [14]. This system consists of six heat exchangers, three flows f_i coming in at different temperatures $\mathbf{t}_i$. The functions of the system are described by polynomial constraints, coming from three kinds of balance:

$\sum_i f_i = 0$: mass balance at each node,
$\sum_i f_i * t_i = 0$: thermal balance at each node,
$\sum_{in} f_i * t_i - \sum_{out} f_j * t_j = 0$: enthalpic balance for each heat exchanger.

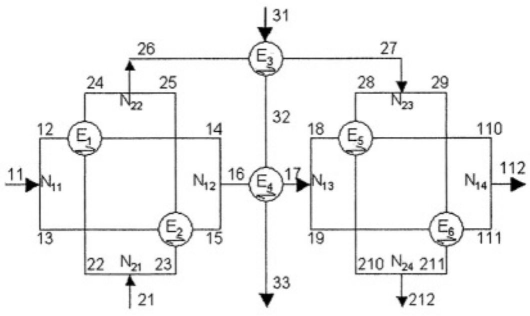


Fig. 1. System of Heat Exchangers

The system has 34 polynomial equations and 54 variables, from which 28 are observable: $t_{11}, t_{12}, t_{13}, t_{16}, t_{17}, t_{18}, t_{19}, t_{112}, t_{21}, t_{26}, t_{27}, t_{212}, t_{31}, t_{33}, f_{11}, f_{12}, f_{13}, f_{16}, f_{17}, f_{18}, f_{19}, f_{112}, f_{21}, f_{26}, f_{27}, f_{212}, f_{31}$ and f_{33}. There is no direct measure of the rest of the variables. This defines three different subsystems, each one formed by two exchangers: $\{E_1, E_2\}$, $\{E_3, E_4\}$ and $\{E_5, E_6\}$. Each of the six exchangers and each of the eight nodes of the system are considered as components whose correct functioning must be verified.

4 Computing All Possible Minimal Conflicts

The model which reflects the system structure and behavior is presented by a set of polynomial constraints. All this information is stored in a CDB.

The key idea is to generate an equivalent constraints model which has the same solution as the original one, but only with observable variables. In order to produce this model we will use Gröbner bases as symbolic technique.

4.1 Gröbner Bases

Gröbner bases theory is the origin of many symbolic algorithms used to manipulate multiple variable polynomials. It is a generalization of Gauss' elimination of multivariable linear equations and of Euclides' algorithm for one-variable polynomial equations. Gröbner bases has better computational properties than the original system. We can determine if a system can be solved or not.

The main idea is to transform the polynomial constraint set into a standard form for the resolution of problems. Having the set of equality polynomial constraints of the form $P = 0$, Gröbner bases produce an equivalent system $G = 0$ which has the same solution as the original one, but generally easier to be solved.

For our work, we have a function called GröbnerBasis, which calculates Gröbner bases by means of a finite set of polynomial equations (SPM) and a set of observable and non-observable variables.

This function allows building the context network. The signature of GröbnerBasis function looks like this:

GröbnerBasis({Polynomials}, {Observable Variables},
{Non-observable Variables})

Let us consider, for instance, the context represented by $\{N_{12}E_1E_2\}$. *Gröbner Basis* function takes the parameters:

GröbnerBasis($\{$**polynomialsOf**$(N_{12}, E_1, E_2)\}, \{f_{16}, f_{12}, f_{13}, t_{16}, t_{12}, t_{13}\}, \{f_{14}, f_{15}, f_{22}, f_{23}, f_{24}, t_{14}, t_{15}, t_{22}, t_{23}, t_{24}\}$)

The result would be the system of polynomial constraints: $\{f_{12} + f_{13} - f_{16} = 0\}$.

4.2 Constraint Database Architecture

One of the difficulties in diagnosing a system is handling the information, therefore we have important reasons to use CDBs in model-based diagnosis:

1. By using CDBs, it is possible to add or delete some components when our system changes. In this way, rebuilding the full problem is not necessary.
2. If we do not use a CDB and the execution of the algorithm diagnosis fails, while being executed, we must reexecute the full problem because there is not partial information stored.
3. CDBs allow using the power of SQL in order to query the database and obtain the necessary information.

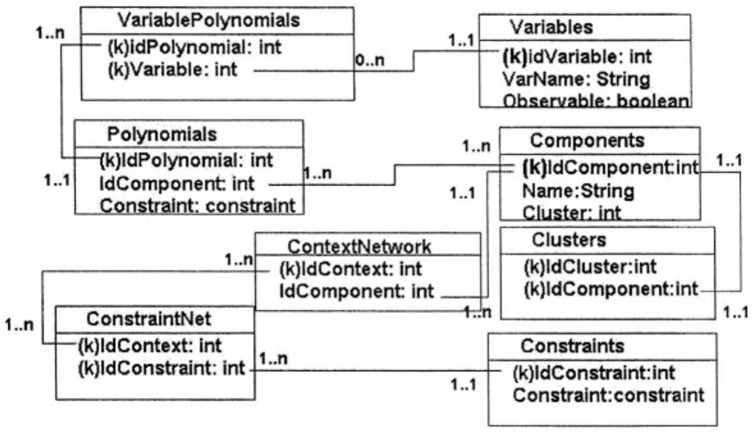


Fig. 2. Constraint Database Architecture (k: Primary Key)

First of all, we are going to explain the database architecture, and how the information is stored:

1. **Components:** This table contains the names and identifiers of the components which make up the system. In this table, the cluster identification of each component is also stored.
2. **Polynomials:** This table contains the different behaviors of the components. The components can have more than one polynomial associated.
3. **ContextNetwork:** This table represents all the relations that the process must study to obtain the minimal possible conflict context.
4. **Variables:** This table contains all the variables which participate in the system, observable and non-observable.
5. **VariablePolynomials:** This table represents the variables in each polynomial. This table is important because in order to obtain Gröbner bases we need to send the observable and non-observable variables of the polynomials.
6. **Constraints:** All the constraints are stored in this table. We will fill in this table with the GröbnerBasis function results.
7. **ConstraintNet:** This table relates each context to constraints.
8. **Clusters:** This table contains the relations between components and clusters.

4.3 First Improvement: Identification of Components Clusters

In our methodology, the first step is to isolate independent subsystems. This structural pretreatment will allow us to divide the system into independent subsystems. The possible minimal conflict sets of the system can be obtained by the conflicts of all independent subsystems. The subsystems obtained are smaller than the whole system. Therefore the computational complexity to detect conflicts from each subsystem is lower or equal than the whole system. In order to clarify the following steps we need the following definition:

Definition 4. Components cluster (CC): A set of components *C* is a components cluster, if the following predicates are true:

- All non-observable inputs and outputs of each component of *C* are always linked only to components of *C*.
- It does not exist another set *C'* with less elements than *C*, which validates the first predicate and it is included in *C*.

With the first predicate we look for the independence among conflicts of different components clusters. This predicate guarantees that it is possible to detect a conflict in a components cluster without information about other components clusters. This is possible because, in a components cluster, all the non-observable inputs and outputs are among components of the same cluster. Therefore, there is not any connection with another components cluster which is not monitored. We look for the division of our system into the biggest possible number of clusters in order to obtain a smaller computational cost. The second predicate guarantees that the components clusters will be as small as possible.

Example: For example, component E_3 is not completely monitored because we are not able to know the value of outputs f_{32} and t_{32}. Likewise, E_4 is not completely monitored because we are not able to know the value of inputs f_{32} and t_{32}. But we can monitor these two components together like they were only one component. In this case, all the inputs and outputs of this component are observable.

Algorithm: The following pseudo-code (see Figure 3) stores the set of components clusters of a system. At first, the set *C* has all the components. The algorithm extracts each time one component of *C* to create an instance *(CP)* of a components cluster. All the components from *C* which have, at least, one non-observable variable in common with one component of *CP* are added to *CP*. If it is impossible to find another component with non-observable variable in common, the components cluster is completed. The process continues with another component from *C* which has not been assigned to any components cluster. The process is finished when the set *C* is empty, and all components are assigned to one components cluster.

```
Set c = ObtainSystemComponents()
while (NotEmpty(c))
   Component x = GetComponent(c)
   Cluster cp = CreateCluster(x)
   boolean change = true
   while(change)
      Set cno = GetCommonComp(cp,c)
      deleteComponents(c,cno)
      AddComponents(cp,cno)
      Change=NotEmpty(cno)
   endwhile
   AddClusterToDB(cp)
endwhile
```

Fig. 3. Pseudocode of the components clusters algorithm

Methods to Select the Components Clusters:

- *ObtainSystemComponents( ):* This method returns all the system components stored in the CDB.
- *GetComponent(Set c):* It returns one component and delete it from the set *C*.
- *CreateCluster(Component x):* Here it is created a cluster with the component *x*.
- *GetCommonComp(Cluster cp, Set c):* This method returns and deletes (from the set *C*) all the components from *c* which have, at least, one non-observable variable in common with some of the components of *cp*.
- *AddComponents(Cluster cp, Set cno):* It adds all *cno* components to *cp* cluster.
- *AddClusterToDB(Cluster cp):* This method stores *cp* Cluster in the CDB.

For the example presented in Section 3, we obtain five components clusters, which are A={{N_{11}}, {N_{13}}, {N_{12},N_{21},N_{22}, E_1,E_2}, {N_{14},N_{23},N_{24},E_5,E_6}, {E_3,E_4}}. For all the components clusters obtained we will build a different and independent context network. With the structural pretreatment the number of nodes is 67. Without this structural pretreatment, the number of nodes of the context network (as it appears in [12]) is 2^{14}-1.

4.4 Second Improvement: Reduction Algorithm

In order to improve the computational time in the calculation of the possible minimal set conflicts, we propose the algorithm of Figure 4. Previously we need two new definitions to understand the algorithm better.

Definition 5. *Observable Context*: It is a context with only one component and, at least, one polynomial without non-observable variables. It means that is not necessary to call GröbnerBasis function. For example $\{N_{11}\}$ and $\{E_3\}$ are observable contexts.

Definition 6. *Relevant Context*: It is a context whose components have, at least, one polynomial whose non-observable variables are also in other polynomial of the context. If we call GröbnerBasis function in other cases, we will not obtain any important results, because it is not possible to eliminate all non-observable variables from, at least, one polynomial of all the context's components:

C is a relevant context if
$\mathbf{C} \equiv \bigcup_i \{c_i\} \mid \forall\, c_i \in \mathbf{C} \cdot \exists\, p_i \in c_i$
$\mid \forall\, \mathbf{x} \in \mathbf{NonObsVar}(p_i) \cdot \mathbf{x} \in \mathbf{C}$
where c_i **is a component,** p_i **a polynomial and NonObsVar(**p_i**)**
the set of non-observable variable of p_i

```
foreach (cluster in clusters)
   Set contexts=ObtainContexts(cluster)
   foreach(context in contexts)
      if(IsAnObservableContext(context))
         AddContext(context)
         UpdateTables(context)
      else
         if (RelevantContext(context))
            AddContext(context)
            CallGröbner(context)
         endif
      endif
   endforeach
endfor
```

Fig. 4. Pseudocode of the reduction algorithm for Relevant Contexts

Methods of the Reduction Algorithm:

- *ObtainContexts(Cluster cluster):* This function returns all the contexts of *cluster*
- *IsAnObservableContext(Context c):* This function returns true if the *context* is an observable context.
- *AddContext(Context c):* This function adds the context *c* to table ContextNetwork .
- *UpdateTables(Context c):* This function stores all the polynomial constraints of the context *c,* in the tables ConstraintNet and Constraint. Here it is not necessary to call GröbnerBasis because the polynomials do not have any non-observable variables.

Table 1. CARCs

Index	CC	Constraints
1	1	f11 - f12 - f13
2	1	-f11 t11 + f12 t12 + f11 t13 - f12 t13
3	2	f17 - f18 - f19
4	2	-f17 t17 + f18 t18 + f17 t19 - f18 t19
5	3	f12 + f13 - f16
6	3	f21 - f26
7	3	f12 t12 + f13 t13 - f12 t16 - f13 t16 + f21 t21 - f21 t26
8	4	f18 + f19 - f112
9	4	f27 - f212
10	4	f18 t18 + f19 t19 - f18 t112 - f19 t112 + f27 t27 - f27 t212
11	5	f26 - f27
12	5	f16 - f17
13	5	f31 - f33
14	5	f16 t16 - f17 t17 + f26 t26 - f27 t27 + f31 t31 - f31 t33

- *RelevantContext(Context c):* This function returns true if the context *c* is a relevant context.

Example: ***Context:*** N_{22}***,*** E_1 ***and*** E_2
In this case the component E_1 *do not have any polynomial with all non-observable variable couple with other component*

To implement this idea, we propose a query to know what are the non-observable variables of a *polynomial* **p** which are also in the same *context* **c** but in different polynomial.

```
SELECT DISTINCT v.VARNAME
FROM VARIABLES v, VARIABLES v2,
     VARIABLEPOLYNOMIALS cv, POLYNOMIALS c,
     VARIABLEPOLYNOMIALS cv2, POLYNOMIALS c2,
     CONTEXTNETWORK rc, CONTEXTNETWORK rc2,
WHERE c.ID=p AND
     c.IDCOMPONENT=rc.IDCOMPONENT AND
     rc.ID=c AND c.ID=cv.ID
     AND cv.VARIABLE=v.IDVARIABLE AND
     v.OBSERVABLE=false AND c.ID<>c2.ID AND
     rc2.ID=rc.ID AND c2.ID=cv2.ID AND
     c2.IDCOMPONENT=rc2.IDCOMPONENT AND

     cv.VARIABLE=cv2.VARIABLE
```

Comparing the query result and the non-observable variable of the polynomial, we will know if all the non-observable variables of a polynomial are in another polynomial, and therefore if it is a relevant context.
- *CallGröbner( ):* We build GröbnerBasis function call with information from the tables ContextNetwork, Polynomials, VariablePolinomial and Components. The results will be stored, if they are not in the table Constraints. Finally, the constraint and the corresponding context will be stored in the table CostraintNet.

Table 2. Improvement using components cluster and relevant context

	No reduction	Using CC	Using CC and RC
Number of Contexts	2^{14}-1	67	67
Calls to GB. function	2^{14}-1	67	7
Obtained Constraints	64	14	14
Elapsed time	4'2 days	7 Seconds	1 Second

(This test have been carried out in a Pentium IV-2Ghz with 512 MB)

With our solution, we only create 11 contexts and we only call GröbnerBasis function 7 times, because the contexts $\{N_{11}\}$, $\{N_{13}\}$, $\{E_3\}$ and $\{E_4\}$ are observable contexts. The reduced algorithm obtains 14 constraints which are shown on Table 1. Table 2 shows the differences among using all contexts (as in [12]) and using our approach.

4.5 Determination of Possible Minimal Conflict Contexts

In order to determinate the possible minimal conflicts we apply a constraint-driven algorithm. The following definition is necessary in this process:

Definition 7. Context Analytical Redundancy Constraint (CARC): It is a constraint derived from SPM, in such a way that only the observable variables are related.

In our approach, the set of CARCs of the system (Table 1) is the union of all the constraints. These constraints were obtained in each components cluster in two ways, directly from observable context or using GröbnerBasis function to each relevant context.

All the relevant context are shown in Figure 5 for the Heat Exchangers example.

5 Conclusions and Future Works

This paper proposes a new approach to automate and to improve the determination of possible minimal conflict sets. The determination of components clusters of the system reduces the number of contexts to consider. Only the relevant contexts are studied in order to reduce the computational complexity. In this paper we propose a CDB architecture to store polynomial constraints using standard SQL and Java^{TM} language to obtain and handle the constraint information. Another advantage is the power of SQL storing and getting information in CDBs.

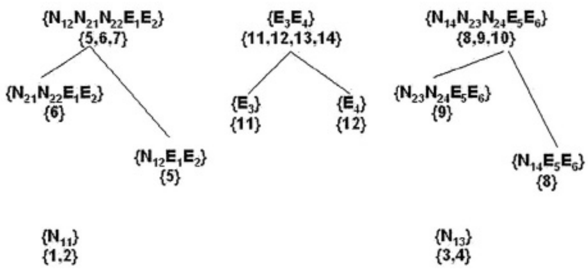

$\{N_{12}N_{21}N_{22}E_1E_2\}$ {5,6,7}
$\{N_{21}N_{22}E_1E_2\}$ {6}
$\{N_{12}E_1E_2\}$ {5}
$\{N_{11}\}$ {1,2}
$\{E_3E_4\}$ {11,12,13,14}
$\{E_3\}$ {11}
$\{E_4\}$ {12}
$\{N_{14}N_{23}N_{24}E_5E_6\}$ {8,9,10}
$\{N_{23}N_{24}E_5E_6\}$ {9}
$\{N_{14}E_5E_6\}$ {8}
$\{N_{13}\}$ {3,4}

Fig. 5. Possible Minimal Conflict Network of the System

Extension to ODEs with polynomial constraints, in order to deal with dynamic systems, is our next objective. At the same time it is interesting for future works to study how the minimal context network changes when some polynomials change, and to look for techniques to avoid restudying all the system. In other way, there is a wide field to study the diagnosis of systems with components are located in different CDBs.

Acknowledgements

This work has been funded by the M. de Ciencia y **Tecnología** of Spanish (DPI2003-07146-C02-01) and the European Regional Development Fund (ERDF/ FEDER).

References

1. Davis, R.: Diagnostic reasoning based on structure and behavior. In Artificial Intelligence 24 (1984) 347–410
2. Reiter, R.: A theory of diagnosis from first principles. Artificial Intelligence 32 **1** (1987) 57–96
3. Kleer, J.D., Mackworth, A., Reiter, R.: Characterizing diagnoses and systems. Artificial Intelligence 56 **2-3** (1992) 197–222
4. Goldin, D., Kanellakis, P.: Constraint query algebras constraints. Journal E. F. editor (1996)
5. P. C. Kanellakis, G.M.K., Revesz, P.Z.: Constraint query languages. Symposium on Principles of Database Systems (1990) 299–313
6. Revesz, P.: Introduction to Constraint Databases. Springer (2001)
7. Kleer, J.D., Williams, B.: Diagnosing multiple faults. Art. Int. (1987)
8. Kleer, J.D.: An assumption-based truth maintenance system. Artificial Intelligence 28 **2** (1986) 127–161
9. Buchberger, B.: Gröbner bases: An algorithmic method in polynomial ideal theory. Multidimensional Systems Theory, N. K. Bose, ed. (1985) 184–232
10. de la Banda, M.G., Stuckey, P., Wazny, J.: Finding all minimal unsatisfiable subsets. Proc. Of the 5th ACM Sigplan Internacional (2003)
11. Frisk, E.: Residual generator design for non-linear, polynomial systems - a gröbner basis approach. In Proc. IFAC Safeprocess, Budapest (2000)

12. Gasca, R., Valle, C.D., Ceballos, R., Toro, M.: An integration of fdi and dx approaches to polinomial models. 14th International Workshop on principles of Diagnosis - DX (2003)
13. Pulido, J.: Posibles conflictos como alternativa al registro de dependencias en línea para el diagnóstico de sistemas continuos. PhD. degree, Universidad de Valladolid (2000)
14. Guernez, C., Petitot, M., Cassar, J., Staroswiecki, M.: Fault detection and isolation on non linear polynomial systems. 15th IMACS World Congress (1997)

Implementation of a Linguistic Fuzzy Relational Neural Network for Detecting Pathologies by Infant Cry Recognition

Israel Suaste-Rivas[1], Orion F. Reyes-Galviz[2],
Alejandro Diaz-Mendez[3], and Carlos A. Reyes-Garcia[4]

[1,3,4] Instituto Nacional de Astrofisica Optica y Electronica (INAOE),
Luis E. Erro 1 Tonantzintla,
Puebla, Mexico

[2] Instituto Tecnologico de Apizaco,
Av. Tecnologico S/N,
Apizaco, Tlaxcala, Mexico

Abstract. In this paper we describe the implementation of a fuzzy relational neural network model. In the model, the input features are represented by their respective fuzzy membership values to linguistic properties. The weights of the connections between input and output nodes are described in terms of their fuzzy relations. The output values during training are obtained with the max-min composition, and are given in terms of fuzzy class membership values. The learning algorithm used is a modified version of the back-propagation algorithm. The system is tested on an infant cry classification problem, in which the objective is to identify pathologies like deafness and asphyxia in recently born babies. The design and implementation of the classifier is presented, as well as results of some experiments.

1 Introduction

The implementation we describe in this article corresponds to the model of a general pattern classifier based on a neural network architecture which uses fuzzy sets in the form of linguistic properties for both input/output data and the structure of the classifier itself. The general system architecture, and the use of Nn-dimensional vectors to represent the fuzzy membership values of the input to linguistic properties was inspired by the work presented by Pal in [1, 2]. The idea of using a relational neural network as a pattern classifier was taken from the general classifier presented by Pedrycz in [3]. The implemented sistem has been tested on an infant cry recognition problem. The pathological diseases in infants are commonly detected several months, and often times, years, after the infant is born. If any of these diseases would have been detected earlier, they could have been attended and maybe avoided by the opportune application of treatments and therapies. It has been found that the infant's cry has much information on its sound wave. Given that the crying in babies is a primary communication function, governed directly by the brain, any alteration on the normal functioning of the babies' body is reflected in the cry. Based on the information contained inside the cry's wave, it can determined the infant's physical state; and even detect physical pathologies, mainly from the brain, in very early phases [4].

2 The Infant Cry Recognition Process

The Automatic Infant Cry Recognition (AICR) process is basically a problem of pattern processing, similar to Automatic Speech Recognition (ASR). In AICR the goal is to take the crying wave as the input pattern, and at the end to obtain the class of cry the vector corresponds to. Generally, the whole process can be divided into two phases. The first phase is known as *signal processing,* or *acoustic feature extraction.* The second phase is known as *pattern classification* or *pattern recognition,* as shown in Figure 1.

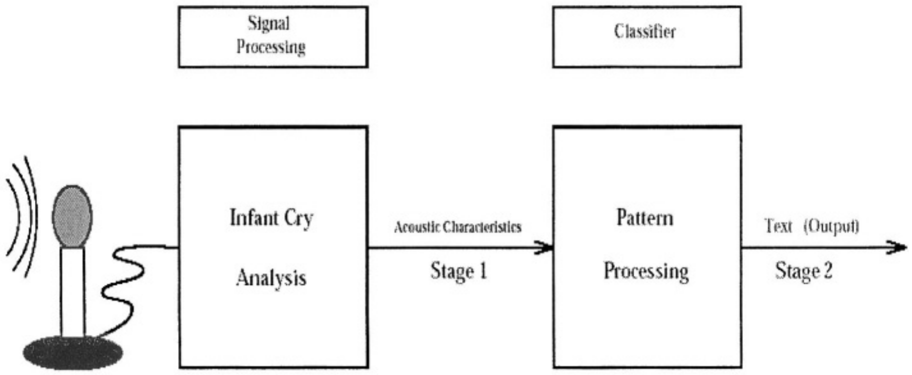


Fig. 1. The Automatic Infant Cry Recognition Process

2.1 Signal Processing

The analysis of the raw cry waveform provides the information needed for its recognition. At the same time, it discards unwanted information such as background noise, and channel distortion [6]. Acoustic feature extraction is a transformation of measured data into pattern data. Some of the most important techniques used for analyzing cry wave signals are [5, 7]: Discrete Fourier Transform (DFT), cepstral processing, and Linear Prediction Analysis (LPA). The application of these techniques during signal processing results in the production of values of a set of acoustic features. The features may be spectral coefficients, linear prediction coefficients (LPC), cepstral coefficients, Mel frequency cepstral coefficients (MFCC), or amplitudes among others [5]. The set of values for

n features may be represented by a vector in an n-dimensional space. Each vector represents a pattern.

2.2 Pattern Classification

During the second phase of the infant cry recognition process, the goal is usually to determine the class or category of each pattern. In this work we classify the patterns by means of a hybrid connectionist model.

3 The Fuzzy Neural Network Model

Pal [1] has suggested that in order to enable a system to handle real-life situations, the concept of fuzzy sets should be incorporated into the neural network, and, that the increase in the amount of computation required, in some cases, with the incorporation of fuzzy logic, is offset by the potential for parallel computation with high flexibility that fuzzy neural networks have. Most of the existing fuzzy neural network models only partially use the elements of fuzzy set theory and techniques, either in the learning process or in their structure [3]. The system proposed in this work is based upon fuzzy set operations in both the structure of the neural network and in the learning process. Following Pal's idea of a general recognizer [2], the model is divided in two main parts, one for learning and another for processing, as shown in Figure 2.

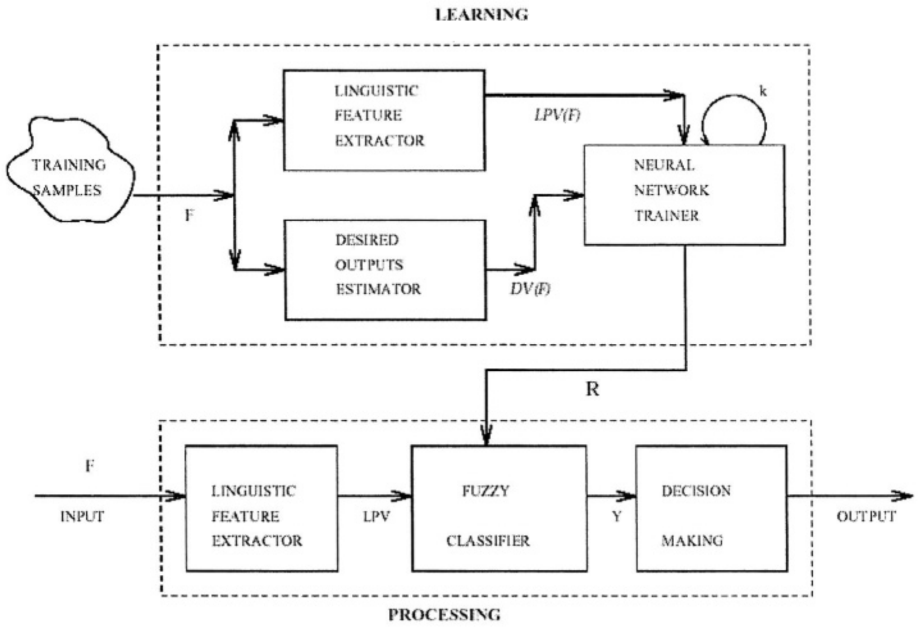


Fig. 2. General Architecture of the Automatic Infant Cry Recognition System

3.1 Fuzzy Learning

The fuzzy learning section is composed by three modules, namely the Linguistic Feature Extractor (LFE), the Desired Output Estimator (DOE), and the Neural Network Trainer (NNT). The learning process starts with the input of the training samples. The Linguistic Feature Extractor takes training samples in the form of n-dimensional vectors containing n features, and converts them to a *Nn* dimensional form vectors, where *N* is the number of linguistic properties. In case the linguistic properties are *low, medium,* and *high,* the resulting $3n$-dimensional vector is called Linguistic Properties Vector (LPV). In this way an input pattern $\mathbf{F}_i = [F_{i1}, F_{i2}, ..., F_{in}]$ containing n features, may be represented as [2]:

$$\begin{aligned}\mathbf{F}_i = & [\mu_{low(F_{i1})}(\mathbf{F}_i), \mu_{med(F_{i1})}(\mathbf{F}_i), \\ & \mu_{high(Fi1)}(\mathbf{F}_i), \ldots, \mu_{high(F_{in})}(\mathbf{F}_i)]\end{aligned} \tag{1}$$

The DOE takes each vector from the training samples and calculates its membership to class k, in an l-class problem domain. The vector containing the class membership values is called the Desired Vector (DV). Both LPV and DV vectors are used by the neural Network Trainer (NNT), which takes them as the bases for training the network. The neural network consists of two layers, the input layer and the output layer. The input layer is formed by a set of *Nn* neurons, with each of them corresponding to one of the linguistic properties assigned to the n input features. In the output layer there are l neurons, where each node corresponds to one of the l classes; in this implementation, each class represents one type of crying. There is a link from every node in the input layer to every node in the output layer. All the connections are described by means of fuzzy relations $R : X \times Y \longrightarrow [0, 1]$ between the input and output nodes. The input vector is clamped to the input layer and the desired output vector is clamped to the output layer during training. The outputs of the network are computed to obtain the error at the output layer. The error is represented by the distance between the actual ouput and the target or desired output. The objective of the training process is to minimize this error. During each learning step, once the error has been computed, the trainer adjusts the relationship values or weights of the corresponding connections, either until a minimum error is obtained or a given number of iterations is completed. The output of the NNT, after the learning process, is a fuzzy relational matrix (*R* in Figure **??**.) containing the knowledge needed to further map the unknown input vectors to their corresponding class during the classification process.

3.2 Fuzzy Processing

The fuzzy processing section is formed by three different modules, namely the Linguistic Feature Extractor (LFE), the Fuzzy Classifier (FC), and the Decision Making Module (DMM). The LFE works in exactly the same way as the one in the learning phase. It is used to calculate the corresponding membership value of each input feature in the classifying vector to each of the linguistic properties. The output of this module is an LPV vector. The LPV vector, along with the fuzzy relational matrix, are used by the Fuzzy Classifier, which obtains the actual outputs from the neural network. The classifier applies the max-min composition to calculate the output. The ouput of this module is an output vector containing the membership values of the input vector to each of the classes. Finally, the Decision Making module takes the values coming from the classifier, and after applying some decision criteria assigns the corresponding class to the testing vector. The assigned class, in this implementation, represents one kind of infant cry.

3.3 Membership Functions

A fuzzy set is defined by a function that maps objects in a domain to their membership value in the set. Such a function is called the *membership function* [9]. In many cases it is recomended to use standard functions whose parameters may be adjusted to fit a specified membership function in an approximate fashion. In the reported experiment the *trapezoidal* membership function was used. There is some evidence that shows that the use of more linguistic properties to describe a pattern point makes a model more accurate [8]. One possibility is the use of seven linguistic properties: *very low, low, more or less low, medium, more or less high, high, very high.*

3.4 Desired Membership Values

Before defining the output membership function, we define the equation to calculate the weighted distance of the training pattern $\mathbf{F}_j$ to the *k*th class in an *l*-class problem domain as in [1]

$$z_{ik} = \sqrt{\sum_{j=1}^{n} \left[\frac{F_{ij} - o_{kj}}{v_{kj}} \right]^2} ,: for\ k = 1, \ldots, l \tag{2}$$

where F_{ij} is the *j*th feature of the *i*th pattern vector, k is the *k*th class. o_{kj} denotes the mean, and v_{kj} denotes the standard deviation of the *j*th feature for the *k*th class. The membership value of the *i*th pattern to class k is defined as follows

$$\mu_k(\mathbf{F}_i) = \frac{1}{1 + (\frac{z_{ik}}{f_d})^{f_e}} ,: \mu_k(\mathbf{F}_i) \in [0, 1] \tag{3}$$

where f_e is the exponential fuzzy generator, and f_d is the denominational fuzzy generator controlling the amount of fuzzines in this class-membership set. In this case, the higher the distance of the pattern from a class, the lower its membership to that class. Since the training data have fuzzy class boundaries, a pattern point may belong to one or more classes in the input feature space.

3.5 The Neural Network Trainer

Neural networks and fuzzy systems estimate input-output functions. Unlike statistical estimators, they estimate a function without a mathematical model. They learn from experience with numerical and, sometimes, linguistic data. Supervised neural networks can learn new patterns and recall old ones simultaneously. Supervised feedforward models provide the most tractable, most applied neural networks. Fuzzy systems store banks of fuzzy associations or common sense rules. They *reason* with parallel associative inference using fuzzy or multivalued sets instead of bivalent propositions. A fuzzy system may infer and modify adaptively its fuzzy associations from numerical samples. In this case, neural and fuzzy systems naturally combine resembling an adaptive system with sensory and cognitive components [10]. The neural network model discussed here is based on the fuzzy neural structure proposed by Pedrycz in [3]. As was previously mentioned, the model works exclusively with set-thoretic operations.

The Relational Neural Network. Let $\mathbf{X} = \{x_1, x_2, \ldots, x_n\}$ be a finite set of input nodes and let $\mathbf{Y} = \{y_1, y_2, \ldots, y_l\}$ represent the set of output nodes in an l-class problem domain. When the max-min composition operator denoted $X \circ R$ is applied to a fuzzy set X and a fuzzy relation R, the output is a new fuzzy set Y, we have

$$Y = X \circ R \tag{4}$$

$$Y(y_j) = max_{x_i}(min(X(x_i), R(x_i, y_j)))$$

where X is a fuzzy set, Y is the resulting fuzzy set and R is a fuzzy relation $R : X \times Y \longrightarrow [0,1]$ describing all relationships between input and output nodes.

Using a vector and matrix notation to represent the max-min composition we have

$$[X(x_1), X(x_2), ..., X(x_n)] \circ$$

$$\begin{bmatrix} R(x_1, y_1) \ldots R(x_1, y_j) \ldots R(x_1, y_l) \\ R(x_2, y_1) \ldots R(x_2, y_j) \ldots R(x_2, y_l) \\ R(x_n, y_1) \ldots R(x_n, y_j) \ldots R(x_n, y_l) \end{bmatrix}$$

We will take the whole neural network represented by expression (8) as a collection of l separate n-input single-output cells. In order to provide the single cell in our model with a rough analogy to the conventional neuron, a threshold or bias $v \in [0,1]$ is incorporated. In this way, the entire unit interval of admissible values at the output node is obtained. The single element is described by

$$Y(y) = max_{x_i}[max(min(X(x_i), R(x_i, y_j))), v(y)] \tag{5}$$

Learning in a Fuzzy Neural Network. If the actual response from the network does not matches the target pattern, the network is corrected by modifying the link weights to reduce the difference between the observed and target patterns. For the relational neural network Pedrycz [3] defines a performance index called equality index, which is

$$T(y) \equiv Y(y) = \begin{cases} 1 + T(y) - Y(y), \text{if } Y(y) > T(y) \\ 1 + Y(y) - T(y), \text{if } Y(y) < T(y) \\ 1, \quad\quad\quad\quad\quad \text{if } Y(y) = T(y) \end{cases} \tag{6}$$

where $T(y)$ is the target output at node y, and $Y(y)$ is the actual output at the same node. $\overline{T}$ is the complement of T defined by $\overline{T}(y) = 1 - T(y)$. In a problem with n input patterns, there are n input-output pairs (x_{ij}, t_i) where t_i is the target value when the input is $\mathbf{X}_{ij}$.

Parameters Updating. In [3] Pedricz discusses the learning scheme for the structure of a neural network with n-inputs and single output, and proposes to complete the process of learning separately for each output node. The updating procedure is made independent of the size of the training set. The learning algorithm is a version of the gradient-descent-based backpropagation algorithm.

Lets consider an n-input-l-output neural network having the following form

$$y_i = f(\mathbf{x}_i; \mathbf{a}, v) = (\bigvee_{j=1}^{n} (a_j \wedge x_{ij})) \vee v$$

where $\mathbf{a} = [a_1, a_2, \ldots, a_l]$ is a vector containing all the weights or relations, $\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{in}]$ is the vector with values observed in the input nodes. The parameters a and v are updated iteratively by taking increments $\triangle a_m$ and $\triangle v_m$ resulting from deviations between all pairs y_i and t_i as follows

$$a(k+1) = a(k) + \Psi_1(k)\left[\frac{\triangle a(k+1)}{Nn} + \eta\frac{\triangle a(k)}{Nn}\right] \tag{7}$$

$$v_m(k+1) = v(k) + \Psi_2(k)\left[\frac{\triangle v(k+1)}{Nn} + \eta\frac{\triangle v(k)}{Nn}\right] \tag{8}$$

where k is the iteration or learning step. Ψ_1 and Ψ_2 are non-increasing functions of k controlling the decreasing influence of increments $\triangle a_m$ and $\triangle v_m$. Ψ is the learning momentum specifying the level of modification of the learning parameters with regard to their values in the previous learning step k. A way of determining the increments $\triangle a_m$ and $\triangle v_m$ is with regard to the mth coordinates of a, $m = 1,2,\ldots,l$.The computation of the overall performance index, and the derivatives to calculate the increments for each cordinate of a, and v are explained in detail in [11].

Once the training has been terminated, the output of the trainer is the updated relational matrix, which will contain the knowledge needed to map unknown patterns to their corresponding classes.

4 Implementation and Results

In the first place, the infant cries are collected by recordings obtained directly by medical doctors. Later, each signal wave is divided in segments of 1 second; these segments are labeled with a pre-established code, and each one constitutes a sample. For the present experiments we have a corpus of 1049 samples of normal infant cry, 879 of hypo acoustics, and 340 with asphyxia. At the following step the samples are processed one by one extracting their MFCC, by the use of the freeware program Praat 4.0 [12]. The acoustic characteristics are extracted as follows: for every second we extract 16 coefficients from each 50-millisecond frame, generating vectors with 304 coefficients by sample. This vectors are further reduced, to a desired size, by the applicacion of PCA. The neural network and the training algorithm are implemented with Matlab. In order to make the training and recognition test, we select 250 samples randomly on each class. From them, 200 samples of each class are randomly selected for training. The training is made up to 10 epochs. After the network is trained, we test it with the 50 samples of each class set apart from the original 250 samples. The recognition accuracy percentage, from each experiment, is presented in a confusion matrix. Different sets of values for the parametres η and k have been tested, along with different initial values for the relational matrix and the bias vector. The best results at present have been obtained with the following parameters $\eta = 0.2$ and $k = 40$. The initial values for the relational matrix and the bias vectors were set as 0.8 in both cases. And the number of input features per vector, after the application of PCA, equal to 10 The feature space was divided in 7 linguistic terms, which makes the dimensionality of the input vector equal to 70.

4.1 Preliminary Results

The results of the model when using the above mentioned set of values, *trapezoidal* membership functions, and considering only the highest membership value of each input pattern in the output vector, are as given in the confusion matrix in Table 4.1

Table 4.1 Confusion Matrix with trapezoidal membership functions

Class	# of Samples	Normal	Deaf	Asphyxia	Accuracy
Normal	50	50	0	0	
Deaf	50	0	50	0	
Asphyxia	50	0	3	47	
Total	150				98 %

Table 4.2 represents the confusion matrix showing the results obtained when using the same set of values, but this time with *gaussian* membership functions. The results are very similar.

Table 4.2 Confusion Matrix with gaussian membership functions

Class	# of Samples	Normal	Deaf	Asphyxia	Accuracy
Normal	50	47	3	0	
Deaf	50	0	50	0	
Asphyxia	50	0	1	49	
Total	150				97.3 %

4.2 Performance Comparison with Other Models

Taco Ekkel [13] tried to classify sounds of newborn cry in categories called normal and abnormal (hypoxia), and reports a result of correct classification of around 85% based on a neural network of radial base. In [14] Reyes and Orozco classify cry samples only from deaf and normal infants, obtaining recognition results that go from 79.05% up to 97.43%.

5 Further Work

Trying to improve the learning and recognition efficiency of the two layer relational neural network, we are in the process of testing the system with some significant changes to its components. One of the major changes is theautomatic optimization of the required parameters, by the use of genetic algorithms. Another important addition is the use of the square relational product as in [15] and [16] besides the circlet product during the recognition phase to generate another criterion to help in the recognition decision.

6 Conclusions

Given the simplicity of the model implementation, the results obtained to date are very encouraging. With the use of the linguistic properties, the impreciseness of the infant cry wave information is reduced by the generation of seven subregions for each acoustic feature, and the use of the max-min composition operation facilitates implementation of the learning task and its performance. Based on the observed results, we are convinced that by implementing the proposed changes the improved fuzzy relational neural network model can provide an effective, reliable and compact infant cry recognizer.

Acknowledgments

This work is part of a project that is being financed by CONACYT-Mexico (37914-A). We want to thank Dr. Edgar M. Garcia-Tamayo and Dr. Emilio Arch-Tirado for their invaluable collaboration when helping us to collect the crying samples, as well as their wise advice.

References

1. Pal, S.K., “Multilayer Perceptron, Fuzzy Sets, and Classification,” in *IEEE Trans. on Neural Networks,* vol 3, No 5, Sep 1992, pp 683-697.
2. Pal, S.K. and Mandal, D.P., “Linguistic Recognition Systems Based on Approximated Reasoning,” in *Information Science,* vol 61, No 2, 1992, pp 135-161.
3. Pedrycz, W., “Neurocomputations in Relational Systems,” in *IEEE Trans. on Pattern Analysis and Mach. Intelligence,* vol 13, No 3, Mar 91, pp 289-296.

4. O. Wasz-Hockert, J. Lind, V. Vuorenkoski, T. Partanen y E. Valanne, El Llanto en el Lactante y su Significación Diagnóstica, Cientifico-Medica, Barcelona, 1970.
5. Ainsworth, W.A., *Speech Recognition by Machine,* Peter Peregrinus Ltd., London, 1988.
6. Levinson S.E., and Roe, D.B., "A Perspective on Speech Recognition," in *IEEE Communications Magazine,* Jan 1990, pp 28–34.
7. Schafer, R.W., and Rabiner, L.R., "Digital Representations of Speech Signals," in *Readings in Speech Recognition,* Morgan Kauffmann Publishers Inc., San Mateo, Calif, 1990, pp 49–64.
8. Park, D., Cae, Z., and Kandel, A., "Investigations on the Applicability of Fuzzy Inference," in *Fuzzy Sets and Systems,* vol 49, 1992, pp 151-169.
9. Yen, J., and Langari, R., *Fuzzy Logic: Intelligence, Control and Information,* Prentice Hall, New Jersey, 1999.
10. Kosko, B., *Neural Networks and Fuzzy Systems,* Prentice Hall, New Jersey, 1992.
11. Reyes, C.A., "On the Design of a Fuzzy Relational Neural Network for Automatic Speech Recognition", Doctoral Dissertation, The Florida State University, Tallahassee, Fl., Apr 94.
12. Boersma, P., Weenink, D. Praat v. 4.0.8. A system for doing phonetics by computer. In-stitute of Phonetic Sciences of the University of Amsterdam. February, 2002
13. Ekkel, T, *Neural Network-Based Classification of Cries from Infants Suffering from Hypoxia-Related CNS Damage,* Master Thesis. University of Twente, The Netherlands, 2002.
14. Orozco Garcia, J., Reyes Garcia, C.A. (2003), "Mel-Frequency Cepstrum Coeficients Extraction from Infant Cry for Classification of Normal and Pathological Cry with Feed-forward Neural Networks", in *proceedings of ESANN 2003,* Bruges, Belgium.
15. Bandler, W. and Kohout, L.J., "Mathematical Relations," in Sing, Mandan G., editor, *Systems & Control Encyclopedia,* Pergamon Press, Oxford, 1987, pp 4000-4008.
16. Bandler, W. and Kohout, L.J., "Fuzzy Relational Products as a tool for analysis and artificial systems," in P.P. Wang, and S.K. Chang, editors, *Fuzzy Sets: Theory and Applications to Policy Analysis and Information Systems,* Plenum Press, New York and London, 1980, pp 341-367.

Adding Personality to Chatterbots Using the Persona-AIML Architecture

Adjamir M. Galvão, Flávia A. Barros, André M. M. Neves, and Geber L. Ramalho

Centro de Informática, Universidade Federal de Pernambuco Caixa Postal 7851, CEP 50732-970, Recife (PE), Brazil

{ }

Abstract. Recent studies highlight the importance of personality for improving human-machine interaction. Attempts of including personality in chatterbots have not been satisfactory regarding the coherence of the chatterbot's behavior, the flexibility and reusability of the personality model. This work presents Persona-AIML, an original architecture for the creation of chatterbots in AIML with personality. It is a flexible architecture that allows the use of different models of personality, described in terms of five elements: traits, attitudes, mood, emotions and physical states. Recent experiments validate the reusability and extensibility of our architecture to build chatterbots with different personalities, however using the same categories base. The implemented chatterbots demonstrated a satisfactory level of coherence in their behavior.

1 Introduction

Chatterbots are computer systems which communicate with people in Natural Language. Since Turing proposed the Imitation Game [1], several attempts of constructing such systems can be identified, from ELIZA [2] to ALICE [3]. Besides facilitating the process of human computer interaction, chatterbots are also able to explore and influence the user's behavior [4]. Recent studies have shown the importance of personality for improving the performance of computer systems [5, 6, 7] (for example, in decision making [8]). Hence, it would be of great interest to construct chatterbots capable of showing personality during interactions with users.

It is possible to identify some attempts of building chatterbots with personality (e.g., ELIZA, JULIA [9]). However, they are not satisfactory concerning the reusability of the personality model for the creation of bots with different personalities. In fact, in these systems, personality is wired in the input-output dialog patterns, requiring significant changes in the dialog base to instantiate a new personality. Also, in some cases, the chatterbot does not show coherence of behavior in the long run.

This work presents Persona-AIML [10, 11], an architecture for the creation of chatterbots with personality in AIML (Artificial Intelligence Markup Language) [3]. Here, chatterbots are modeled as rational agents [12], and use a rule base to describe their personality. This is a flexible architecture that allows the use of different models of personality in the construction of chatterbots.

The current prototype contains a rule base with personality elements, and a categories base with input-output dialog patterns in AIML. The first personality model investigated was based on the Five Factors Model (also known as The Big Five) [13], where personality can be composed by the following elements: traits, attitudes, mood, emotions and physical states. Recent experiments validate the reusability and extensibility of our architecture to build chatterbots with different personalities, however based on the same personality model and using the same categories base. The Persona-AIML chatterbots used in the experiments demonstrated a satisfactory level of coherence in their behavior.

We highlight this work as relevant and original. We did not find in the literature chatterbots with a flexible component for the development of personality, allowing for the creation of bots with different personalities based on the same model, or based on different models of personality. We only found ad-hoc implementations without a satisfactory level of coherence in the chatterbot's behavior.

Section 2 presents a brief discussion about computational models of personality. Section 3 presents the Persona-AIML architecture in detail, and section 4 shows recent experiments. Section 5 presents conclusions and future work.

2 Personality in Chatterbots

Although the construction of chatterbots was initially focused on academic studies, the growth of the Web stimulated the use of these systems as virtual assistants in different applications (e.g., entertainment, electronic commerce, and distance learning). As these systems are aimed to interact with users in natural language, personality turns out to be a central issue in their design, in order to improve their performance by exploring and influencing the user's behavior.

Among the analyzed chatterbots with personality, we highlight ELIZA and JULIA. ELIZA was designed to behave as a Rogerian therapist, and its technology influenced the development of bots for decades (some of which participated in the Loebner Prize[1], one of the most important Chatterbots competitions). Despite their success, ELIZA's like chatterbots have their personality embedded in the input-output dialog patterns, what degrades the flexibility and reuse of the dialog base in the creation of different models of personality.

JULIA, on the other hand, does not bear a clear personality model like ELIZA. She just tries to make the user believe that she is human (e.g., insisting that she is female and getting angry when a user says that she is a computer). As in ELIZA, here personality is also embedded in the dialog's patterns. In fact, none of the analyzed systems offers components for the flexible creation of personality models (see [10, 11] for further details).

[1] http://www.loebner.net/Prizef/loebner-prize.html

2.1 Models of Personality

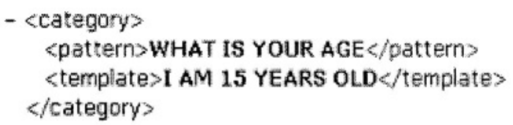
According to Krech and Crutchfield [14], the definition of personality in Psychology considers several aspects (e.g., traits, beliefs, attitudes, emotional reactions, mood), also indicating the way in which these elements are organized and related.

Computational models of personality are, in general, adapted from some Psychology model or theory. However, the definition of these computational models is sometimes difficult, since the adaptation of theories is not always trivial (or even feasible). The following theories and models have been used in the development of computational models of personality: Theory of Traits [15]; the Five Factors Model [13]; Theory of Social Learning [16]; and the OCC Model [17].

Several computational models of personality for intelligent agents have been proposed (e.g., [4, 5, 6, 7, 18, 19]), each one following a particular psychological personality model. Regardless the adopted personality model, in general, a computational model can be defined in terms of two aspects [14]: (1) a set of personality elements (e.g., mood and emotions); and (2) the way these elements interact and intervene with the decision making process. If the agent's architecture deals with these aspects as separate components, the developer can customize them to any particular personality model. Clearly, a modular architecture adds flexibility to the process of creating agents with personality.

2.2 The AIML Language

AIML (Artificial Intelligence Markup Language) is a markup language based on XML (extensible Markup Language) for the construction of chatterbots. Our analysis of existing technologies pointed at AIML as one of the most adequate language for the development of these bots in the Internet. Although AIML does not offer a component for the creation of personality models, it presents other desired characteristics, such as portability, open source and memory of the dialog (allowing the control of the dialog context). Chatterbots in AIML maintain a knowledge base of categories (pairs of input-output patterns - see Fig. 1).



```
- <category>
    <pattern>WHAT IS YOUR AGE</pattern>
    <template>I AM 15 YEARS OLD</template>
  </category>
```

Fig. 1. Example of AIML category

In order to ensure a high performance in dialogs with users, AIML's bases are in general very large (ALICE's current base contains about 30.000 categories). Due to the lack of components for dealing with personality, personality elements have to be embedded within the categories. This practice compromises the reusability of the categories base (a central component of the chatterbot). Yet, the construction of chatterbots with sophisticated and coherent personality may be unfeasible, since this solution is not scalable to large category bases.

3 The Persona-AIML Architecture

Taking into account the requirements presented in the previous section, we developed the Persona-AIML Architecture for the construction of personality models in AIML chatterbots. Below we present the design decision that guided the development of Persona-AIML (section 3.1), followed by a detailed discussion of its components (section. 3.2).

3.1 Design Decisions

Chatterbots developed in AIML follow an architecture where the reasoning process (perceive-reason-act) is integrated into the available categories. The AIML interpreter receives the user sentence as input (perceptive data) and identifies its category (reasoning). Following, the interpreter returns to the user the output template of the selected category (action). Clearly, this approach does not favor the modeling of personality since, in order to define the chatterbot's behavior, the botmaster (the chatterbot's creator and/or manager) must embed personality elements within the categories in the AIML base. Considering the size of AIML bases, this is not a straightforward task. Yet, to create a chatterbot with a different personality, it is necessary to rewrite the categories base.

The Persona-AIML was designed as an extension of the original AIML technology, since we consider it adequate for the development of chatterbots in the Internet. Our main aim was to favor the integration of personality elements into existing AIML bots[2].

However, due to the drawbacks of this language (mentioned above), some design decisions have been made. First of all, our architecture follows the intelligent agents model described in [12], which allows the separation of the reasoning sequence perceive-reason-act. As such, this architecture allows the construction of reactive agents with internal state, providing a framework for the development of chatterbots with a coherent personality in the long run.

Persona-AIML was designed within the symbolic paradigm, which, we believe, offers more flexibility for modeling chatterbots personality. Besides, the symbolic approach is largely used in the literature in the implementation of synthetic actors [8] (which are closely related to chatterbots, since they also simulate "illusion of life").

3.2 Architecture's Components

Based on the above decisions, Persona-AIML counts on four components (Fig. 2): the Categories Base (with AIML dialog categories), the Dialog Log (which registers the current dialog), the Personality Component (with beliefs, personality elements, and rules that determine the chatterbot's behavior), and the Reasoning Component.

One of our main concerns was to provide an architecture that would allow the use of different personality models. This is made possible by the Personality

[2] There exists around 50,000 AIML bots currently running - see http://alicebot.org

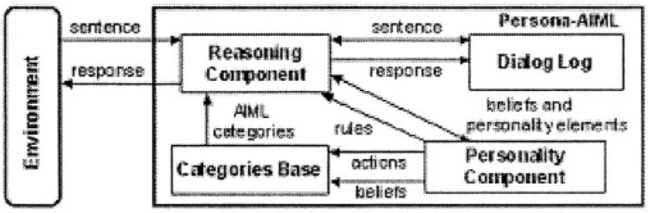


Fig. 2. Persona-AIML Architecture

Component. The Reasoning Component identifies the user's sentence category, verifies the rules in the Personality Component, and then determines the response to be returned to the user, also considering the context of the dialog. Responses in the Personality Component rules have priority over the responses in the Categories Base.

Categories Base. In the Persona-AIML Categories Base, each category must inform its purpose (e.g., a question or a greeting), indicated in the tag ¡pattern-type¿. This information is used by the Personality Component to determine the response to the user. Fig. 3 presents a Persona-AIML category that represents a question to the chatterbot. The default response is defined in tag ¡do¿, and it is returned to the user only when no other response is selected by the Reasoning Component based on personality rules.

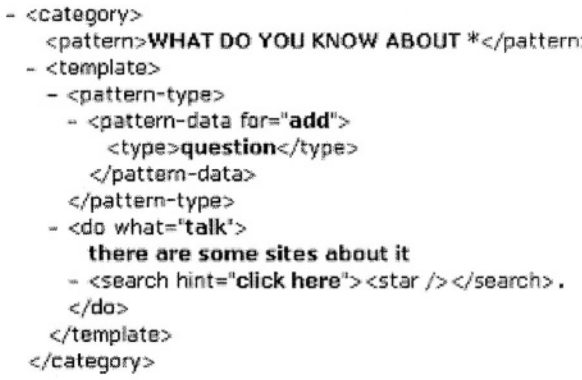
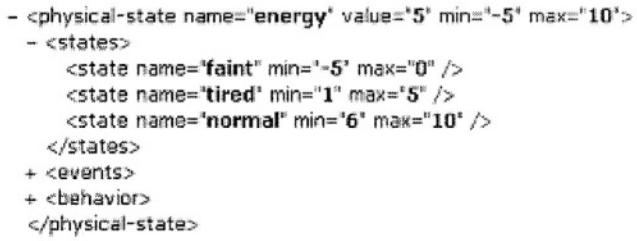

```
- <category>
    <pattern>WHAT DO YOU KNOW ABOUT *</pattern>
  - <template>
    - <pattern-type>
      - <pattern-data for="add">
          <type>question</type>
        </pattern-data>
      </pattern-type>
    - <do what="talk">
        there are some sites about it
      - <search hint="click here"><star /></search>.
      </do>
    </template>
  </category>
```

Fig. 3. Example of a Persona-AIML category

Personality Component. In order to allow the construction of different personality models, this component offers a set of personality elements and a validation mechanism that controls the use of these elements by the botmaster. It contains a knowledge base of facts and if-then rules. The facts represent values associated to the personality elements, and the chatterbot's beliefs, which contain information about the chatterbot (e.g., name and age). These beliefs can only be modified by the rules in this component. The rules can be of two types: meta-rules (to control the inference engine's priorities), and ordinary rules used to represent personality elements. This knowledge base is created the by bot-master.

Personality Elements. The Persona-AIML architecture allows the use of the following personality elements: attitudes, emotions, mood, physical states and traits. These elements are represented by facts and if-then rules that determine the chatterbot's personality. The current intensity of each element is represented by a fact (e.g., [mood = 3] means that chatterbot is sad). Fig. 4 presents the definition of the physical state *energy.*



```
- <physical-state name="energy" value="5" min="-5" max="10">
  - <states>
      <state name="faint" min="-5" max="0" />
      <state name="tired" min="1" max="5" />
      <state name="normal" min="6" max="10" />
    </states>
  + <events>
  + <behavior>
  </physical-state>
```

Fig. 4. Example of a personality element

Two types of rules can be defined. The rules of events are verified when the chatterbot's internal state changes (e.g., if the chatterbot is sad then it cries). Behavior rules are verified during the construction of the response (e.g., if the user compliments the chatterbot, then it thanks the user). Both types of rules can embed relations between two or more elements (e.g., if the chatterbot likes the user [attitude] then it becomes happy when 'talking' to this user [emotion]). These two types of rules can be triggered in the same dialog turn. However, event rules are always verified in the first place, since the bot's the response depends on its internal state.

The personality elements used here were chosen based on studies of the existing computational models (section 2), as well as on the works of Frijda [20] and Brave and Nass [21]. However, different from these models, the elements in our architecture present the following characteristics: (1) rules can be directly associated to personality elements, thus facilitating the construction and maintenance of the chatterbot's rule base; (2) the intensity of each element can be defined by an expression including other elements; and (3) specific rules for attitudes and emotions can be constructed for a particular user or group of users (e.g., teachers in a distance learning environment).

Validation Mechanism. The (high-level) personality model adopted by the bot-master specifies the personality elements (and their relations) to be considered in the chatterbot's construction. The corresponding computational model must follow this guideline. This model is implemented by the Validation Mechanism, which verifies the coherence between the considered set of personality elements and its relations during the creation of the Personality Component rules. This mechanism must guarantee that the elements considered by personality model are the only ones appearing in the base. For example, if the personality model considers only emotions and traits, the mechanism must verify whether other elements were used (e.g., attitudes), reporting this incoherence to the botmaster.

This mechanism must also guarantee that the relations in the rules respect the adopted personality model. For example, if a trait is defined as "steady", the botmaster is able to identify the occurrence of some invalid rule by verifying whether there are rules that modify the intensity of that trait.

Reasoning Component. The Reasoning Component determines the response to be returned to the user based on the chatterbot's personality, also taking into account the dialog context. Two processes are executed in this component (Fig. 5): Pattern Matching and Response Construction.

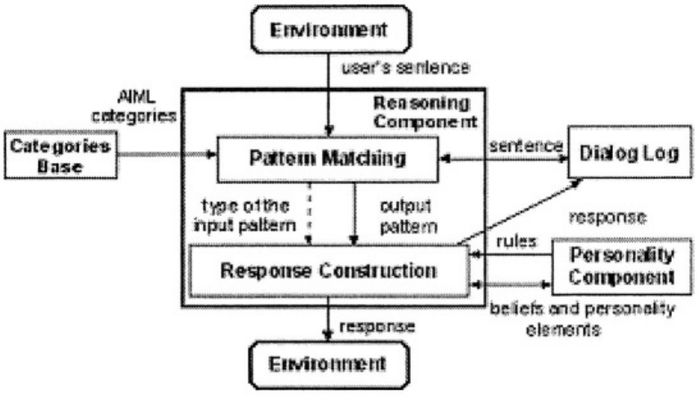


Fig. 5. Reasoning Component

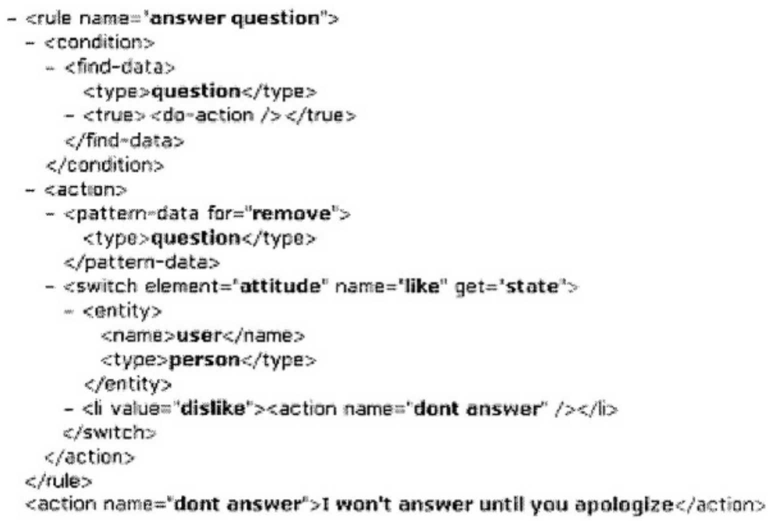
The first process determines the category that better matches the user's sentence, considering the dialog's context. The selected category provides the type of the input sentence and the default output pattern to be returned to the user in case no other response is selected during the response construction process. The response construction includes the update of the chatterbot's internal state and the decision making. This process verifies the rules of behavior in the Personality Component, allowing the execution of more than one rule in the same dialog turn. This way, the response to the user is chosen based on the chatterbot's personality, also taking into account the dialog context.

As said, the responses in the Personality Component rules have priority over the responses in the Categories Base. This mechanism guarantees independence between the Categories Base and the personality model used in the construction of the chatterbot, favoring extensibility and reusability of this base.

4 Experiments and Results

Initial experiments [10, 11] with the Persona-AIML architecture were based on a chatterbot which used a Categories Base in Portuguese (the Pixelbot)[3]. An analysis of dialog logs revealed that this bot was capable of changing mood and behavior when " attacked" by the user. In order to validate the reusability and extensibility of our architecture, as well as to obtain more significant results, we performed new experiments, described below.

In all experiments, the implemented Personality Component follows the Big Five model, based on the Theory of Traits. According to this theory, personality is defined by traits that determine the personality elements intensity. Fig. 6 shows an example of a behavior rule and action considered in the prototype.



```
- <rule name='answer question'>
  - <condition>
    - <find-data>
        <type>question</type>
    - <true> <do-action /> </true>
      </find-data>
    </condition>
  - <action>
    - <pattern-data for="remove">
        <type>question</type>
      </pattern-data>
    - <switch element='attitude' name='like' get='state'>
      - <entity>
          <name>user</name>
          <type>person</type>
        </entity>
      - <li value='dislike'><action name='dont answer' /></li>
      </switch>
    </action>
  </rule>
  <action name='dont answer'>I won't answer until you apologize</action>
```

Fig. 6. Example of a behavior rule and action

In recent experiments, three different chatterbots were used: one without an explicit personality model (Bot-1, the original Pixelbot), and two others with different personalities (Bot-2 and Bot-3). These three bots use the same Categories Base. Bot-2 and Bot-3 use different personality rules (however based on the same model): Bot-2 is more patient with the user, starting a dialog session with the attitude "like" (the user) and with a high introversion/extroversion trait. Bot-3, in turn, does not "like" the user from the start, and have a low introversion/extroversion trait.

In order to assess the chatterbots' performance, we asked ten users to maintain dialog sessions with the three bots without knowing in advance which was

[3] http://www.virtus.ufpe.br, http://150.161.189.230/virtusbot/index.htm

Table 1. Experiments results

Form summary	Bot-1	Bot-2	Bot-3
Any change of mood was noticed?	40%	80%	80%
What was the most interesting bot?	30%	60%	100%
The bot's behavior was coherent to the programmed personality?		10%	80%

Bot-1, Bot-2 or Bot-3. The users were asked to fill in a form based on these dialogs. Table 1 summarizes some experiments' results.

These results show that the users were able to notice more changes of mood (a personality element) in the Persona-AIML chatterbots. It is important to say that Bot-1 (the original Pixelbot) has a few "reactions" embedded in some categories (e.g., to become angry when attacked by the user). However, these changes of mood do not always occur when expected by the users (see table 1), and are not always coherent (according to users' reports). On the other hand, Bots 2 and 3 showed a high level of changes in mood during the dialogs, as expected by the users. Over 50% of the users considered Bot-2, a Persona-AIML bot, the most interesting one. Finally, we observe that Bot-3 was considered the least interesting one. As pointed by the users, this is due to the fact that Bot-3 is the most impatient one.

The third question in Table 1 was not directly asked to the users. Actually, we asked them to describe the personality of each bot and interpreted their answers. All users were able to correctly identify the personality of Bot-2, whereas 80% of them correctly identified the personality of Bot-3. These results indicate that our bots were able to show a coherent behavior during the dialogs with users.

These experiments validated the reusability and extensibility of our architecture since the three bots exhibited a different behavior, however using the same Categories Base. Besides, Bot-2 and Bot-3 used the same personality model, however bearing different personalities.

5 Conclusions and Future Work

This work presents Persona-AIML, an original architecture for the development of chatterbots with personality. Our main aim was to fill in a gap in this research area providing a flexible architecture that allows the creation of different and coherent personality models. Moreover, Persona-AIML favors the reuse of AIML bases, since the botmaster needs only to include the Personality Component in order to create bots with personality. Tests with the prototype showed the effectiveness of the proposed architecture. The main contributions of this work are: the development of a generic architecture for the creation of personality models and the integration of personality in AIML chatterbots.

As future work we can propose: (1) the development of other models of personality using the Persona-AIML; (2) the creation of chatterbots for environments that involve other entities besides the user (e.g., virtual reality environments); (3) to verify whether and how the chatterbot can influence the user's behavior; (4) the construction of a personality editor that would facilitate the definition of the chatterbot personality.

References

[1] Turing, A.: Computing Machinery and Intelligence. Mind. Vol. 59 (236),(1950) 433-460

[2] Weizenbaum, J.: Eliza - A Computer Program for the Study of Natural Language Communication Between Man and Machine. Communications of the ACM. Vol. 9, (1966) 36-45

[3] Wallace, R.: Don't Read Me - A.L.I.C.E. and AIML Documentation. http://www.alicebot.org/dont.html (2001)

[4] Moon, Y.: Intimate Self-Disclosure Exchanges: Using Computers to Build Reciprocal Relationships with Consumers. Harvard Business School, MA (1998)

[5] Elliot, C.: The Affective Reasoner: A Process Model of Emotions in a Multi-agent System. Ph.D. Thesis, Technical Report no. 32, Institute for the Learning Sciences, Northwestern University, Evanston, IL (1992)

[6] Reilly, W.: Believable Social and Emotional Agents. PhD Thesis. School of Computer Science, Carnegie Mellon University (1996)

[7] Oliveira, E. and Sarmento, L.: Emotional Valence-Based Mechanisms and Agent Personality. In: LNAI, Vol. 2507. Springer-Verlag (2002) 152-162

[8] Silva, D.: Atores Sinétticos em Jogos de Aventura Interativos: O Projeto Enigmas no Campus. MSc. Thesis, Universidade Federal de Pernambuco, Brazil (2000)

[9] Mauldin, M.: Chatterbots, Tinymuds, and the Turing Test: Entering the Loebner Prize Competition. In Proceedings of the Twelfth National Conference on Artificial Intelligence. American Association for Artificial Intelligence (1994)

[10] Galvão, A.: Persona-AIML: Uma Arquitetura para Desenvolver Chatterbots com Personalidade. MSc. Thesis, Centre de Informática, Brazil (2003)

[11] Galvão, A.; Neves, A.; Barros, F.: Persona-AIML: Uma Arquitetura para Desenvolver Chatterbots com Personalidade. In.: IV Encontro Nacional de Inteligência Artificial. Anais do XXIII Congresso SBC. Vol.7. Campinas, Brazil (2003) 435-444

[12] Russel, S. and Norvig, P.: Artificial Intelligence: a Modern Approach. Prentice-Hall, Englewood Cliffs (1995)

[13] Mccrae, R. and John, O.: An Introduction to the Five-Factor Model and its Applications. Journal of Personality, Vol. 60 (1992) 175-213

[14] Krech, D. and Crutchfield, R.: Elementos da Psicologia. Pioneira, So Paulo, Brazil (1973)

[15] Allport, G.: Traits Revisited. American Psychologist, 21, 1-10 (1966)

[16] Bandura, A.: Social Learning Theory. Prentice-Hall, Englewood Cliffs (1977)

[17] Ortony, A., Clore, A. and Collins, G.: The Cognitive Structure of Emotions. Cambridge University Press, Cambridge, England (1988)

[18] Rousseau, D. and Hayes-Roth, B.: A Social-Psychological Model for Synthetic Actors. Technical Report KSL-9707. Knowledge Systems Laboratory, Stanford University (1997)

[19] Kshirsagar, S., Magnenat-Thalmann, N.: Virtual Humans Personified. In Proceedings of the First AAMAS International Joint Conference. ACM Press (2002) 356-357

[20] Frijda, N.: Varieties of Affect: Emotions and Episodes, Moods, and Sentiments. In: P. Ekman and R. J. Davidson (eds.): The Nature of Emotion. Oxford, University Press, New York (1994) 59-67

[21] Brave, S. and Nass, C.: Emotion in Human-Computer Interaction. In: J. Jacko and A. Sears, (eds.): Handbook of Human-Computer Interaction. Lawrence Erlbaum Associates, New York (2002) 251-271

DIMEx100: A New Phonetic and Speech Corpus for Mexican Spanish

Luis A. Pineda[a], Luis Villaseñor Pineda[b], Javier Cuétara[c], Hayde Castellanos[a,c], and Ivonne López[a,c]

[a] Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas (IIMAS, UNAM)
luis@leibniz.iimas.unam.mx,
{haydecv, ivlom}@turing.iimas.unam.mx
[b] Instituto Nacional de Astrofísica, Optica y Electrónica (INAOE)
villasen@inaoep.mx
[c] Facultad de Filosofía y Letras, UNAM
j_cuetara@correo.unam.mx

Abstract. In this paper the phonetic and speech corpus DIMEx100 for Mexican Spanish is presented. We discuss both the linguistic motivation and the computational tools employed for the design, collection and transcription of the corpus. The phonetic transcription methodology is based on recent empirical studies proposing a new basic set of allophones and phonological rules for the dialect of the central part of Mexico. These phonological rules have been implemented in a visualization tool that provides the expected phonetic representation of a text, and also a default temporal alignment between the spoken corpus and its phonetic representation. The tools are also used to compute the properties of the corpus and compare these figures with previous work.

1 Introduction

Despite recent progress in speech recognition, the availability of phonetic corpora for the creation of acoustic models in Spanish is still very limited[1]. The creation of this kind of resources is required for a variety of reasons: TTSs (text to speech systems) need to be targeted to specific linguistic communities, and acoustic models for the most common allophones of the dialect need to be considered in order to increase recognition rates. A linguistic and empirically motivated allophonic set is also important for the definition of pronunciation dictionaries. Mexican Spanish, for instance, is a language with 22 phones (17 consonants and 5 vowels), but our empirical work with the dialect of the center of the country has shown that there are 37 allophones (26 consonant sounds and 11 vowels and semi-consonants) that appear

[1] Among the few available corpus we can mention the Latino-40 Spanish Read News for Latin-American Spanish, available LDC (http://www.ldc.upenn.edu), the ALBAYZIN Corpus for Castillan Spanish and the Spanish Speech Corpus 1, these later two available at ELDA (http://www.elda.fr).

often enough in spoken language to be considered in transcriptions and phonetic dictionaries, and whose acoustic properties deserve the definition of an acoustic model for each unit in this set. Previous corpora for Mexican Spanish, like Tlatoa (Kirshning, 2001) and Gamboa (2002) have only considered the main phonemes of the language, and have conflicting criteria for the transcription of some consonants (e.g. *y* in *ayer*) and semi-consonant sounds (e.g. [j] and [w]). Another consideration is that phonetic contexts depend on the dialect, and it is possible to define a set of phonological rules that predict the phonetic units that are likely to be pronounced by a speaker of the dialect in a given context. This information is useful not only for the TTS, but also for the definition of automatic tools that help the transcription process.

In this paper we present the design and collection process of the DIMEx100 corpus, including its linguistic and empirical motivation and the computational methodology and tools employed and developed for its transcription. We also present an assessment of the corpus properties in relation to previous work.

2 Corpus Design and Characteristics

For the collection process we considered the Web as a large enough, complete and balanced, linguistic resource, and selected the corpus phrases from this source; the result of this exercise was the Corpus230 (Villaseñor *et al.,* 2004) consisting of 344,619 phrases with 235,891 lexical units, and about 15 million words. From this original resource we selected 15,000 phrases from 5 to 15 words; these phrases were ordered according to its perplexity value from lowest to highest; intuitively the perplexity is a measure of the number of linguistic units that can follow a reference unit in relation to the corpus: the lower the perplexity of a word, for instance, the less the number of different words that are likely to follow it in a sentence; the perplexity of a sentence can then be defined as a function of the perplexity of its constituent words; accordingly, sentences with a low perplexity are constituted by words with a high discriminating power or information content in relation to the corpus. For the final corpus we took the 7000 sentences with the lowest perplexity value. Phrases with foreign words and unusual abbreviations were edited out, and the set was also edited for facilitating the reading process and for enhancing the relationship between text and sound (e.g. acronyms and numbers were spelled out in full). Finally we arrived at a set of 5010 phrases; the corpus was recorded by 100 speakers, each recorded 50 individual phrases, in addition to 10 phrases that were recorded by all 100 speakers. With this we arrived to 6000 phrases: 5000 different phrases recorded one time and 10 phrases recorded 100 times each. The final resource is the DIMEx100 corpus. In order to measure the appropriateness of the corpus we controlled the characteristics of the speakers; we also measured the amount and distribution of samples for each phonetic unit, and verified that these were complete in relation to our allophonic set and balanced in relation to the language. These figures are presented below in this paper.

The corpus was recorded in a sound study at CCADET, UNAM, with a Single Diaphragm Studio Condenser Microphone Behringe B-1, a Sound Blaster Audigy

Platimun ex (24 bit/96khz/100db SNR) using the Wave Labe program; the sampling format is mono at 16 bits, and the sampling rate is 44.1 khz.

2.1 Socio-linguistic Characteristics of the Speakers

Recording an oral corpus implies considering and designing minimal linguistic measurable aspects in order to be able to evaluate them afterwards. Following Perissinotto (1975), the speakers were selected according to age (16 to 36 years old), educational level (with studies higher than secondary school) and place of origin (Mexico City). A random group of speakers at UNAM (researchers, students, teachers and workers) brought in a high percent of these kind of speakers: the average age was 23.82% years old; most of the speakers were undergraduate (87%) and the rest graduate and most of the speakers (82%) were born and lived in Mexico City. As we accepted everyone interested, 18 people from other places residing in Mexico City participated in the recordings. The corpus resulted gender balanced (49% males; 51% women). Although Mexican Spanish has several dialects (from the northern region, the Gulf Coast and Yucatan's Peninsula, to name only a few) Mexico City's dialect represents the variety spoken by most of the population in the country (Canfield, 1981; Lope Blanch, 1963-1964; Perissinotto, 1975).

2.2 Linguistic Characteristics

From our empirical study (Cuétara, 2004) of the DIME Corpus (Villaseñor *et al.,* 2000), developed within the context of the DIME Project (Pineda *et al.*, 2002), a set of 37 allophones for Mexican Spanish that occur often enough and can be clearly distinguished was identified; this set is being used for the transcription process. This set is illustrated in Table 1. The Mexbet phonetic alphabet (Uraga and Pineda, 2002; Cuétara, 2004) is used for the representation of the phonetic units, but the table should be self explanatory given the articulation point for each allophone and the distinctive features in the left column.

In addition to this allophonic set we have identified the context in which all of these allophones occur. These contexts have been characterized through phonological rules; we present two instances for illustration: the rule for the bilabial voiced stop /b/ and the rule for the open vowel /a/. In table 2 "///_" stands for an initial context; "b_c" stands for the closure of the "b" allophone (which is considered as a unit in phonetic transcriptions), and "V" stands for the fricative realization of /b/. So, the unvoiced stop /b/ occurs in initial contexts, and after /m/ or /n/, and its fricative realization everywhere else. The vowel /a/, in turn, can be realized as a velar in front of other velar sounds and in front of /l/ in the coda of a syllable (e.g. *alto*); this is indicated by the "$" sign in "_{1}$". The palatal /a/, in turn, is realized in front of palatal sounds, and the central /a/ elsewhere. In general, the allophonic variation of Spanish can be modeled with phonological rules (Moreno and Mariño, 1998), and the full set applicable to Mexican Spanish is presented in (Cuétara, 2004).

Table 1. Allophonic set for Mexican Spanish

		Labial	Labio-dental	Dental	Alveo-lar	Palatal					Velar			
Unvoiced stops		p		t		k_j "queso", "kilo"					k			
Voiced stops		b		d							g			
Unvoiced affricate						tS "hacha"								
Voiced africate						dZ "lluvia", "un yunque"								
Unvoiced fricatives			f	s_["asta"	s						x			
Voiced fricatives		V "haba"		D "hada"	z "mismo"	Z "ayer", "mi yunque"					G "haga"			
Nasals		m		n_["antes"	n	n~ "año"					N "angel"			
Liquids	Lateral				l									
	Vibrants				r("pero" r "perro"									
Vowels	High					j "aire"	i						u	w "aura"
	Medium							e				o		
									E "erre"		O "sol"			
	Low								a_j "aire"	a	a_2 "aunque", "alma"			

Table 2. Examples of Phonological Rules

Bilabial voiced stop b		
/b/	b_c b	///_
/b/	b_c b	{m, n}_
/b/	V	Elsewhere

Open vowel a		
/a/	a_2	_{u, x}
/a/	a_2	_{l}$
/a/	a_j	_{tS, n~, Z, j}
/a/	a	Elsewhere

3 Computational Tools

In this section we present three computational tools that we have developed to assist human transcribers in the transcription process:

1. A set of transcription rules mapping textual representations (graphemes) to their corresponding phonological and phonetic representations (i.e. transcriptions).
2. An interactive visualization tool or "phonetizer" that translates a given text into its phonological and phonetic representation using these grapheme to phoneme and grapheme to allophone conversion rules.

3. A script that given a speech signal and its associated textual transcription produces the expected segmental representation with a default temporal alignment between the phonetic and allophonic units and its associated speech signal.

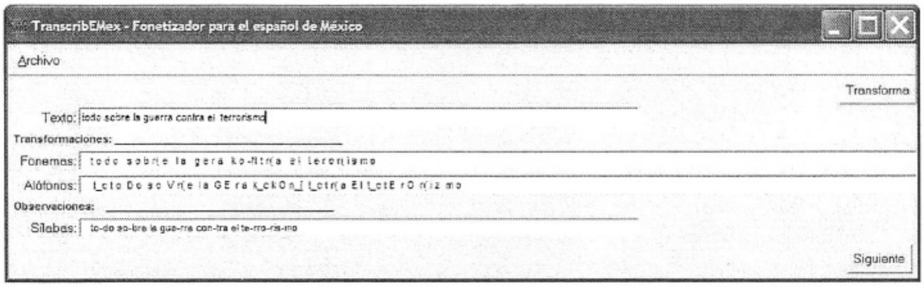
3.1 Textual to Phonetic Representations Translation Rules

For this translation, the textual representation is first divided into syllables. A syllabizer that follows the standard syllabic structure of Spanish has been implemented with standard regular expressions in Perl. The syllabic representation is then read left to right scanning the orthographic symbols one at the time in two stages: first, orthographic symbols are substituted by their corresponding phoneme representation; in Spanish, the mapping from graphemes to phonemes is not one to one: a given symbol can have no phonetic realization (e.g. *h* is never realized, or *u* in the context of *g_e*), and more than one (e.g. *c* as [k] and [s])[3]; also, two symbols can be realized as one sound (e.g. *ch* as [tS], *y* and *ll* as [Z] or [dZ]), etc. We have developed set of rules including one for each possible situations. Orthographic contexts are represented through regular expressions, and substitution rules are applied from the most specific to the most general context. In the second stage, phoneme representations are substituted by their corresponding allophonic representations; for this translation, a regular expression matching the context for every symbol, as shown in Table 2, is defined. Rules are applied from the most specific to the most general context too; for instance, for the orthographic translation of /b/ the rule for initial context is tried first, next the rule for the nasal left context and finally the default context rule that rewrites the fricative realization; similarly for the open vowel *a,* Archiphonemes are also considered: b/p, d/t, g/k, n/m and r(/r have no contrasting value when they appear in the coda of a syllable and both of the elements in each pair are subsumed in the common representation /-B/, /-D/, /-G/, /-N/ and /-R/ respectively (e.g. *optar* is transcribed as /o-Bta-R/ and *camión* as /kamio-N/).

3.2 The Phonetizer

The grapheme to phoneme and allophonic representations were used to build a visualization tool, using Perl and tcl/Tk. With this interface the user can type the text, and the phonemic and allophonic transcriptions are displayed. The tool provides an initial expectation for the human transcriber in the temporal alignment process. It is also useful for didactical purposes and has been used to train human transcribers in the DIME-II Project, and has been received very positively by undergraduate students of phonetic courses at UNAM too. The phonetizer is illustrated in Figure 1.

[2] The *x* grapheme has at least three pronunciations: [ks], [x] and [s]. Currently, we give the standard pronunciation [ks], but we plan to extend the rules with a pronunciation dictionary for Mexican Spanish. We also have cases of grapheme to grapheme conversion: *w* is conventionally rewritten as *gü.*



Texto: todo sobre la guerra contra el terrorismo
Transformaciones:
Fonemas: todo sobr(e la gera ko-Ntr(a el teror(ismo
Alófonos: t_cto Do so Vr(e la GE ra k_ckO n_[t_ctr(a El t_ctE rO r(iz mo
Observaciones:
Sílabas: to-do so-bre la gue-rra con-tra el te-rro-ris-mo

Fig. 1. Phonetic representations visualization tools

3.3 Default Transcription with Temporal Alignment

The transcription rules have also been used to construct a tool that given a speech signal and its orthographic transcription produces a default phonetic transcription temporally aligned with the speech data; mean temporal durations were computed for all allophones from the DIME Corpus, and this figures are continuously updated with the data produced by the human transcription of DIMEx100 itself. We are using the CSLU Tool Kit's SpeechView[3] program for the human transcription, and the default temporal alignment is produced in the format for tags of this program (.phn). This tool facilitates greatly the phonetic transcription task: the human transcriber simple loads the speech and the tag's files, and completes the transcription manually. An illustration of the SpeechView interface, with the speech data and the default tags is shown in Figure 2.

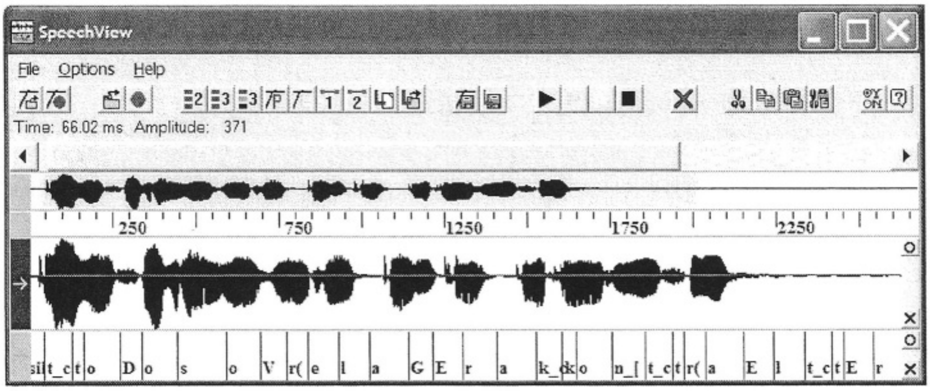

SpeechView
File Options Help
Time: 66.02 ms Amplitude: 371
250
750
1250
1750
2250
sil t_c t o D o s o V r(e l a G E r a k_c k o n_[t_c t r(a E l t_c t E r

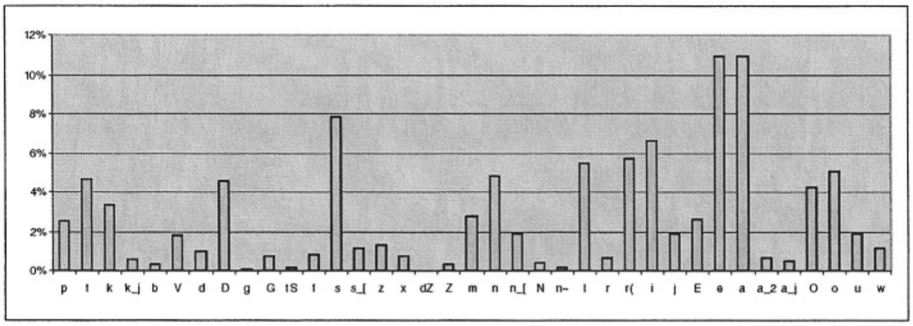
Fig. 2. Default temporal alignments

[3] http://cslu.cse.ogi.edu

4 Corpus Statistics and Evaluation

The text to phonetic representation rules also allow us to evaluate whether the corpus is complete, large enough and balanced. For this we simply translate the text into its phonemic and allophonic representations, and compute the number and distribution of samples; as expected, the corpus includes all phonetics units, with a large enough number of samples for the training process. In particular, the less represented phonetic units are [n~] with 338 samples, [g] with 227 and [dZ] with 23; with the exception of this latter allophone, which can be subsumed with [Z] or complemented with supplementary sample words, the figures are satisfactory. Table 3 shows the expected distribution of the corpus according to rules for all 5010 different phrases; the actual figures for the spoken phrases (i.e. the speech corpus) will be reported when the human transcription is available. Also, the set of phonological rules for all allophones specifies all possible contexts in which these units can occur; the contexts in the right column of Table 2 for all the allophones abstract over all contexts of the form $\alpha\beta\gamma$, where α and γ are the left and right allophones in relation to a reference allophone β. As we have a significant number of instances of all allophones in the corpus, all the contexts specified in the phonological rules appear systematically, and we conclude that the corpus is complete[4]. This is consistent with the perplexity based method used for the corpus design, despite that this computation was performed at the level of the words.

Table 3. Phonetic distribution of DIMEx100



These figures can also be used to assess whether the corpus is balanced. In Table 4 we compare the distribution of DIMEx100 in relation to Quillis (1981), Llisterri and Mariño (1993) for peninsular Spanish and Pérez (2003) for Chilean Spanish. As can be seen in Table 4 our balancing procedure follows closely the figures of previous studies, taken into account the allophonic differences between the dialects, and we

[4] For an alternative balancing method see (Uraga and Gamboa, 2004).

conclude that DIMEx100 is balanced. The correlation at the level of the phonemes between DIMEx100 and these three corpora is shown in Table 5.

Table 4. Phonetic distribution of Spanish

Phones	Alophones	Quilis	Llisterri & Mariño	Pérez	DIMEx100
/p/	[p]	2.77	2.6	2.58	2.57
/t/	[t]	4.53	4.63	4.92	4.66
/k/	[k]	3.98	4.04	3.94	3.33
	[k_j]	-	-	-	0.57
/b/	[b]	2.37	0.45	1.92	0.32
	[V]	-	2.47	-	1.79
/-B/		0.03	-	-	-
/d/	[d]	4.24	0.76	4.84	0.98
	[D]	-	3.2	-	4.59
/-D/		0.31	-	-	-
/g/	[g]	0.94	0.11	0.94	0.09
	[G]	-	0.79	-	0.73
/-G/		0.28	-	-	-
/tS/	[tS]	0.37	0.4	0.32	0.15
/f/	[f]	0.55	0.51	0.75	0.8
/T/	[T]	1.45	1.53	-	-
/s/	[s]	8.32	6.95	9.61	7.86
	[z]	-	1.33	-	1.27
	[s_[]	-	-	-	1.11
/x/	[x]	0.57	0.63	0.74	0.7
/Z/	[Z]	0.41	0.19	0.69	0.31
	[dZ]	-	-	-	0.01
/m/	[m]	3.06	3.63	2.62	2.77
/n/	[n]	2.78	7.02	7.78	4.85
	[n_[]	-	-	-	1.86
	[N]	-	0.46	-	0.38
/-N/		4.86	-	-	-
/n~/	[n~]	0.25	0.27	0.24	0.13
/l/	[l]	4.23	4.25	5.05	5.43
/L/	[L]	0.38	0.54	-	-
/r(/	[r(]	3.26	4.25	6.19	5.7
/r/	[r]	0.43	0.4	0.64	0.62
/-R/		1.93	-	-	-
/i/	[i]	7.38	4.29	7.46	6.6
	[j]	-	2.6	-	1.9
/e/	[e]	14.67	13.73	14.13	10.94
	[E]	-	-	-	2.59
/a/	[a]	12.19	13.43	12.31	10.96
	[a_j]	-	-	-	0.49
	[a_2]	-	-	-	0.62
/o/	[o]	9.98	10.37	9.28	5.05
	[O]	-	-	-	4.26
/u/	[u]	3.33	1.98	3.05	1.87
	[w]	-	1.35	-	1.14

Table 5. Correlation of DIMEx100 with previous corpora

Coeficientes de correlación contra DIMEx100	
Quilis	0.97
Llisterri & Mariño	0.98
Pérez	0.99

5 Conclusions

In this paper we have presented the DIMEx100 corpus for the creation of acoustic models for Mexican Spanish. The design methodology and corpus statistics have also been reported. We have considered the Web as a large enough, complete and balanced linguistic resource and a simple method for the corpus design, based on an entropy measure, was developed; this method was used with very satisfactory results as the resulting corpus is complete and balanced. The transcription, currently under development, is based on a set of allophones and phonological rules that were identified through an empirical investigation of the DIME Corpus. The set of phonetic units and phonological rules has also been used for the development of computational tools to support the transcription process. The main tools are "a phonetizer", that given a text returns its phonemic and allophonic transcription, and a script that given a speech signal and its orthographic transcription returns an allophonic transcription with a default temporal alignment; the first tool has also been used as a tutorial aid in phonetics and phonology, both for the members of the DIME-II Project, and also in formal phonetics courses in the school of Spanish linguistics at UNAM's *Facultad de Filosofía y Letras.* The tools have also used to measure the expected number and distribution of phonetic units in the corpus, and we have been able to assess the corpus in relation to previous work, with very positive results.

Acknowledgements

The corpus DIMEx100 has been developed within the context of the DIME-II Project, at IIMAS, UNAM, with the collaboration of *Facultad de Filosofía y Letras,* UNAM, and INAOE in Tonanzintla, Puebla. The authors wish to thank the enthusiastic participation of all members of the project who participated in the collection and transcription of the corpus (in progress): Fernanda López, Varinia Estrada, Sergio Coria, Patricia Pérez, Iván Moreno and Isabel López; also to the 100 locutors that participated in this effort. We also wish to thank the audio support provided by José Mondragón at CCADET and the computational support provided by César Gamboa at DCC-IIMAS. We also wish to acknowledge the facilities provided by Dr. Felipe Lara and Dr. Fernando Gamboa to record the corpus at CCADET, UNAM. The authors also acknowledge the support of Conacyt's grant 39380-U and LAFMI (Laboratorio Franco Mexicano de Informática).

References

1. Canfield, D. L. (1981/1992). Spanish pronunciation in the Americas, Chicago: The University of Chicago Press.
2. Cuétara, J. (2004). Fonética de la ciudad de México. Aportaciones desde las tecnologías del habla. MSc. Thesis in Spanish Linguistics, UNAM, Mex (In Spanish).
3. Gamboa, C. (2001). Un sistema de reconocimiento de voz para el Español. BSc. Thesis. UNAM, Mex (In Spanish).
4. Kirschning, I. (2001) Research and Development of Speech Technology and Applications for Mexican Spanish at the Tlatoa Group. Development Consortium at CHI 2001, Seattle, WA.
5. Lope Blanch, J. M. 1963-1964/1983. En torno a las vocales caedizas del español mexicano, en Estudios sobre el español de México, México: Universidad Nacional Autónoma de México, pp. 57-77 (In Spanish).
6. Llisterri, J. and Mariño, J. B. (1993). "Spanish adaptation of SAMPA and automatic phonetic transcription". Reporte técnico del ESPRIT PROJECT 6819, Speech Technology Assessment in Multilingual Applications, 9 pp.
7. Moreno, A. and Mariño, J. B. (1998). Spanish dialects: Phonetic transcription, Proceedings of ICSLP'98. The 5th International Conference on Spoken Language Processing, Sydney.
8. Pérez, H. E. (2003). Frecuencia de fonemas, Concepción: Universidad de Conception, Chile. (In Spanish).
9. Perissinotto, G. 1975. Fonología del español hablado en la Ciudad de México. Ensayo de un método sociolingüístico, México: El Colegio de México (In Spanish).
10. Pineda, L. A., Massé, A., Meza, I., Salas, M., Schwarz, E., Uraga, E. and Villaseñor, L. (2002). The DIME Project, Proceedings of MICAI2002, Lectures Notes in Artificial Intelligence, Vol. 2313, Springer-Verlag.
11. Quilis, A. (1981/1988). Fonética Acústica de la Lengua Española, Madrid: Gredos. (In Spanish).
12. Uraga, E. and Pineda, L. A. (2002). Automatic generation of pronunciations lexicons for Spanish, Computational Linguistics and Intelligent Text Processing, Third International Conference CICLing 2002, Alexander Gelbuck (ed.), Lecture Notes in Computer Science, Vol. 2276, Springer-Verlag, pp. 330-339, Berlin.
13. Uraga, E. and Gamboa, C. (2004). VOXMEX Speech Database: design of a phonetically balanced corpus, Fourth International Conference on Language Resources and Evaluation, Lisbon, Portugal, May 2004.
14. Villaseñor, L., Massé, A. and Pineda, L. (2000). A Multimodal Dialogue Contribution Coding Scheme, Proceedings of ISLE workshop, LREC2000 Athens, May 29-30.
15. Villaseñor, L., Montes y Gómez, M., Vaufreydaz, D. and Serignat, J. F. (2004). Experiments on the Construction of a Phonetically Balanced Corpus from the WEB, Proceedings of CICLING2004, LNCS, Vol. 2945, Springer-Verlag.

Author Index

This page intentionally left blank

This page intentionally left blank

This page intentionally left blank